Deliverable D5.4
Report on implementation and testing of security and energy management techniques

Document Summary Information

Grant Agreement No	871780	**Acronym**	MonB5G
Full Title	Report on implementation and testing of security and energy management techniques		
Start Date	01/11/2019	**Duration**	42 months
Project URL	https://www.monb5g.eu/		
Deliverable	D5.4 Report on implementation and testing of security and energy management techniques		
Work Package	WP5		
Contractual due date	M39	**Actual submission date**	31/01/2023
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	BCOM		
Responsible Author	Cao-Thanh PHAN (BCOM), Eric GATEL (BCOM)		

Contributions from	Sabra Ben Saad (EUR), Karim Boutiba (EUR), Mohamed Mekki (EUR), Abderahmane Tlili (EUR), Sofiane Messaoudi (EUR), Adlen Ksentini (EUR), Anestis Dalgkitsis (IQU), Eric Gatel (BCOM), Cao-Thanh PHAN (BCOM), Cédric MORIN (BCOM), Georgia Pantelide (eBOS), George Tsolis (CTXS), Luis Blanco (CTTC), Engin Zeydan (CTTC), Sarang Kahvazadeh (CTTC)
Reviewers	George Tsolis (CTXS), Sabra Ben Saad (EUR), Vasiliki Vlahodimitropoulou (OTE), Kukliński Sławomir (Orange PL)

Executive Summary

Motivated by the increased attack surface of slicing-enabled beyond 5G environments, MonB5G aims at providing a secure and trusted environment for network slice deployment and management. The distributed nature of MonB5G elements and their AI-based features are instrumental in achieving these objectives. By applying advanced data analysis and Machine Learning (ML) techniques, early and accurate identification of attacks will be possible via the automatic application of localized, self-healing mechanisms to mitigate them, taking advantage of the distribution of security management tasks and security enforcement points across the architecture.

This is the last deliverable of Work Package 5. The deliverable presents the implementation and testing of security and energy-aware mechanisms proposed and developed within the context of the tasks 5.1, 5.2 and 5.3. Different scenarios of attacks have been carried out to evaluate the performances of those components of the MonB5G solution (MS-Monitoring System, AE-Analytics Engine, DE- Decision Engine, ACT-Actuator) with different platforms. Once implemented, measurements have been carried out to retrieve relevant metrics and KPI and to measure the performance of those components.

aLTEr attack is a MITM attack type that is carried out between the user equipment (UE) and the gNodeB (gNB) and consists of breaking layer two of the user radio bearer, exploiting the user data integrity protection as a vulnerability to carry out the attack. To conduct the detection and remediation of this attack, a 5G system in SA mode has been deployed on virtualized infrastructures, based on cloud native technology of Kubernetes (K8s) and virtual machines (VM), and different IA tools were used to detect attacks. Metrics such as mean time to detect and mean time to react were considered to evaluate the performances of the solution.

mMTC attack is an attack originating from a subset of User Equipment attached to a specific network slice issuing malicious traffic towards the infrastructure services. A typical example is compromised MTC devices generating a massive number of network attachment requests. This type of attack targets mainly the shared network slice elements, such as the 5G Core Network (CN). For that purpose, another 5G testbed was used and closed-loop control components (i.e. MS, AE, and DE) and an Element Manager (EM) on top of the AMF were implemented to measure, analyse and detect the attack. Relying on machine learning to detect abnormal traffic and perform the mandatory actions as a reaction to this attack, different algorithms were tested and several KPI were computed to measure the performances of those components.

A third scenario has been implemented to secure federated learning in B5G networks, against poisoning attacks. The design of this framework comprises four main steps, starting from generating a realistic dataset to designing a detection scheme of poisoning attacks. A real platform using OAI with 5G radio access and CN, linked to a tool (my5G-rantester) that emulates user equipment, have been built so that MonB5G components could be deployed. Performances of the components were measured in such scenario.

Eventually, from an energy-efficiency perspective, the feasibility and scalability of the proposed stochastic federated learning algorithm was tested through cloud-native FL agents, using a Docker Compose tool to emulate cloud deployments, as it typically runs on top of Kubernetes and is supported by Container

Orchestration Engines (COE) offered as a Platform as a Service (PaaS). Model validation and simulations were used to evaluate the solution and to demonstrate its ability to jointly reduce average service latency by 103.4% and energy consumption by 17.1% compared to other solutions.

These achievements mean that all components are ready to be used for different PoCs and scenarios, which will be the main activities and achievements of Work Package 6.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

ACT *Actuator*

AE *Analytics Engine*

COE *Container Orchestration Engines*

CP *Control Plane*

DB *Data Base*

DE *Decision Engine*

DoT *DNS over TLS*

DSM *Domain Slice Manager*

EARCD *Eurecom AMF Resource Consumption Dataset*

E-Divisive with Medians *EDM*

EM *Element Manager*

FP *False Positive*

FW *Firewall*

IDSM *Inter-Domain Slice Manager*

KNN *k-nearest neighbours*

MAE *Mean Absolute Error*

MCC *Mobile Country Code*

MDP *Markov Decision Process*

MITM *Man In The Middle*

ML *Machine Learning*

MNC *Mobile Network Code*

MS *Monitoring System*

MSE *Mean squared error*, *Mean Squared Error*

PaaS *Platform as a Service*

RMSE *Root Mean Squared Error*

ROC Receiver Operating Characteristic

SO *Service Orchestrator*

SPAN *Switched Port Analyser*

Subscription Permanent Identifier *SUPI*, *SUPI*

TN *True Negative*

VM *virtual machines*

VXLAN *Virtual Extensible LAN*

1. Introduction

1.1 Scope

MonB5G targets the deployment of a novel autonomic management and orchestration framework by heavily leveraging the distribution of operations together with state-of-the-art AI-based mechanisms. The MonB5G approach focuses on the design of a hierarchical, fault-tolerant, automated, and data-driven network management system that incorporates security as well as energy efficiency as key features in order to orchestrate a high number of parallel network slices and significantly higher types of services in an adaptive and *zero-touch* way.

This deliverable is the final report of MonB5G contributions of Work Package 5, dedicated to Security and Energy Enhancement Mechanisms. While the deliverables D5.2 and D5.3 focused respectively on activities towards reaching zero-touch security management relying on AI and activities for a zero-touch distributed x10 energy-efficient MANO compared to a centralized approach, this deliverable (D5.4) presents the implementation and evaluation of those proposed solutions. First, driven by use cases, such as aLTEr attack, mMTC attack, poisoning attack on Federating Learning and components such as Stochastic Federated Learning, the implementation of the MonB5G key components Monitoring System (MS), Analytic Engine (AE), Decision Engine (DE) and Actuator (ACT) is detailed for each scenario. Then, the deliverable details metrics and KPI from real measurements or from simulations. Final goal of the implementation and evaluation is to get all inputs and knowledge to integrate those components in the final platform, let by WP6.

1.2 Structure

The main structure of this deliverable is the following:

- Section 2 explains and describes in detail the implementation of the components of the MonB5G architecture, namely MS (Monitoring System), AE (Analytics Engine), DE (Decision Engine) and ACT (Actuator) components for security uses cases and for algorithms of energy reduction.

- Section 3 deals with the evaluation of the performances of these components for the uses cases and scenarios selected in the project. It describes in detail the methodology, the KPI formulas and the measured values of the KPI used for the evaluation, based on real measurement or simulations.

2. Implementation

2.1 Security

2.1.1 SCENARIO ALTER ATTACK

To conduct the detection and remediation scenario of attack aLTEr[1], we deployed a 5G system in SA mode on virtualized infrastructures based on cloud native technology of Kubernetes (K8s) and virtual machines (VM). Actually, the 5G control plane network functions are K8s services and pods while the user plane, the simulators of the base station gNodeB and the user equipment (UE) are applications running on VMs. In addition to supplying the 5G system, we are deploying a software framework that supports core functions for handling cyber security incidents. These core functions can detect cybersecurity threats and minimize the impact of security incidents. The software framework is designed to allow the software components performing the detection and remediation functions to communicate with each other by means of an event-based communication service. The event-based communication service mutually decouples the event producer from the consumer while allowing them to exchange information in near real-time. Other substantial benefits include ease of scaling, improved robustness, and component adaptability. Figure 1 shows the overview of the platform that has been built, including the incident response module as a security orchestrator, to detect and respond quickly to minimize the impact on the 5G network.

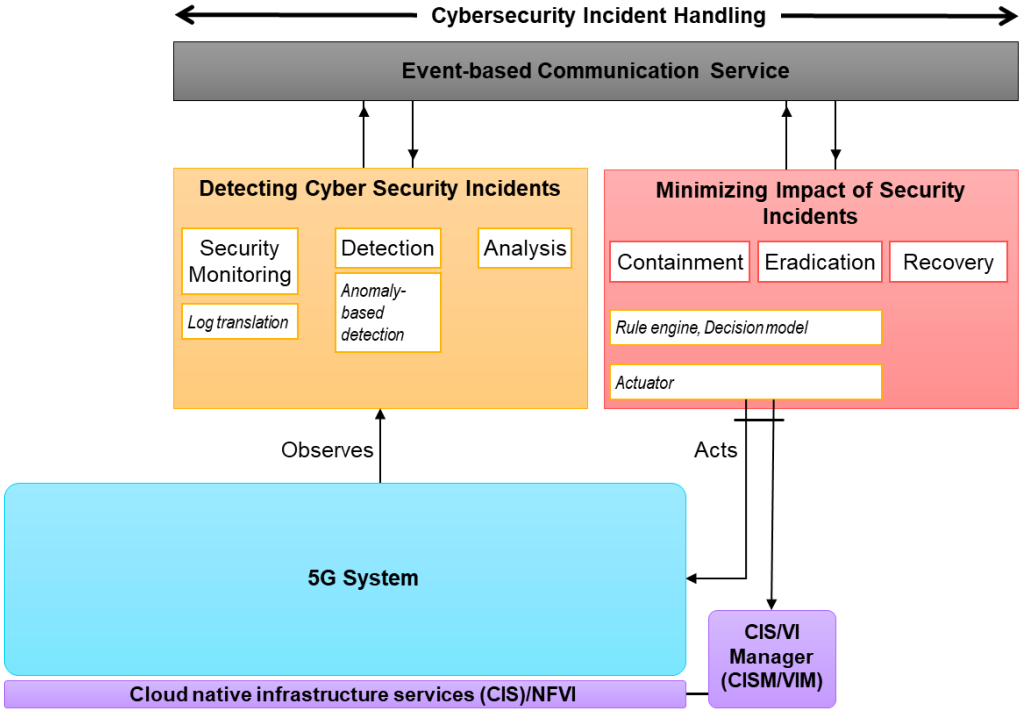


Figure 1: Global cybersecurity platform with 5G SA system and components for security

[1] https://alter-attack.net/

2.1.1.1 IMPLEMENTATION OF THE THREAT DETECTION SUBSYSTEM

The core function threat detection aims at implementing appropriate activities to identify the occurrence of cybersecurity events. It includes the continuous security monitoring of the 5G system and its assets, and the detection of anomalous activities.

For the detection of aLTEr attack, which modus operandi consists in redirecting DNS messages to the malicious server when the integrity protection option has been disabled, our approach consists of observing network activity at the N6 interface in the UPF. This observation of network traffic provides an understanding of the normal behaviour of network capacity utilisation and the usage of protocols, so that any deviation from this expectation can lead to the reporting of a cyber security incident. The MonB5G MS is used to perform the continuous security monitoring of the user plane data at the N6 interface, it is implemented as two network function components, 1/ data source and 2/ collector:

- The data source takes a mirrored copy of the network traffic at the N6 interface in the UPF via a dedicated port called Switched Port Analyser (SPAN), which filters the packets to be collected according to the IP protocols (in our case, the DNS packets) in order to minimise the amount of data it transmits to the collector via a logical tunnel such as a Virtual Extensible LAN (VXLAN) tunnel. In our implementation, the UPF leverages a virtual switch running in the user space of a VM.
- The collector is a K8s service, it processes the raw data it receives from data sources via a tunnel. The collector outputs high level protocol transaction logs, which are a synthetic and condensed representation of the raw data. The protocol transaction logs are sent as monitoring events to the detection and remediation communication service, where they are made available for further processing such as detection of cyber security events. Here, the opensource tool Zeek[2] is used as a network monitor to perform raw data to high level log translation.

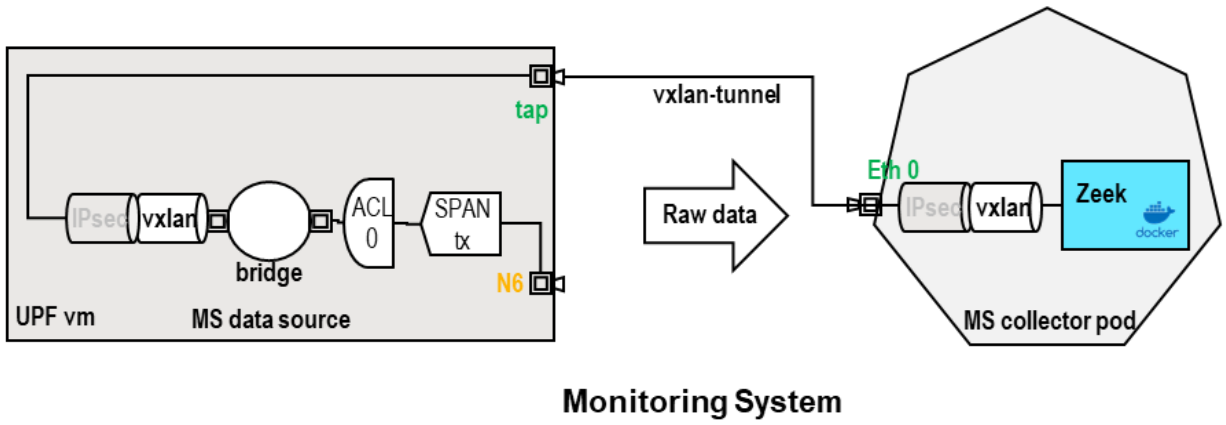


Figure 2: Monitoring system architecture to collect and transform user data for the security monitoring

[2] https://zeek.org/

The second network function component of the core function Threat Detection is the anomaly detection handled by the MonB5G AE. This component parses the logs it receives from the MonB5G MS as events available in a dedicated topic of the event-based communication service as shown in Figure 3. This component embeds an ML (Machine Learning) model of normal network activity that has been pre-trained with malfunction-free data. To show the feasibility of using AI based algorithms in the AE, an anomaly detection algorithm based on ML is proposed to detect the anomalies in the DNS server IP addresses. It is assumed that there is no knowledge about the attack pattern and its characteristics. Thus, as we have no labelled and classified data to train the ML algorithm, an unsupervised ML method is used to do the anomaly detection in the DNS logs. Several unsupervised anomaly detection ML algorithms have been selected for the MonB5G AE; however, it is very challenging to achieve high detection accuracy to avoid producing false positive alerts, therefore the core function detection should validate anomaly events by other operations, such as modifying system settings, obtaining related data from other sources, or manually validating alerts.

If an anomaly event is confirmed as a cyber security incident, the MonB5G AE produces an incident report and sends it in a dedicated topic of the event-based communication system, so that remediation can be carried out to minimize the impact of the incident on the 5G system.

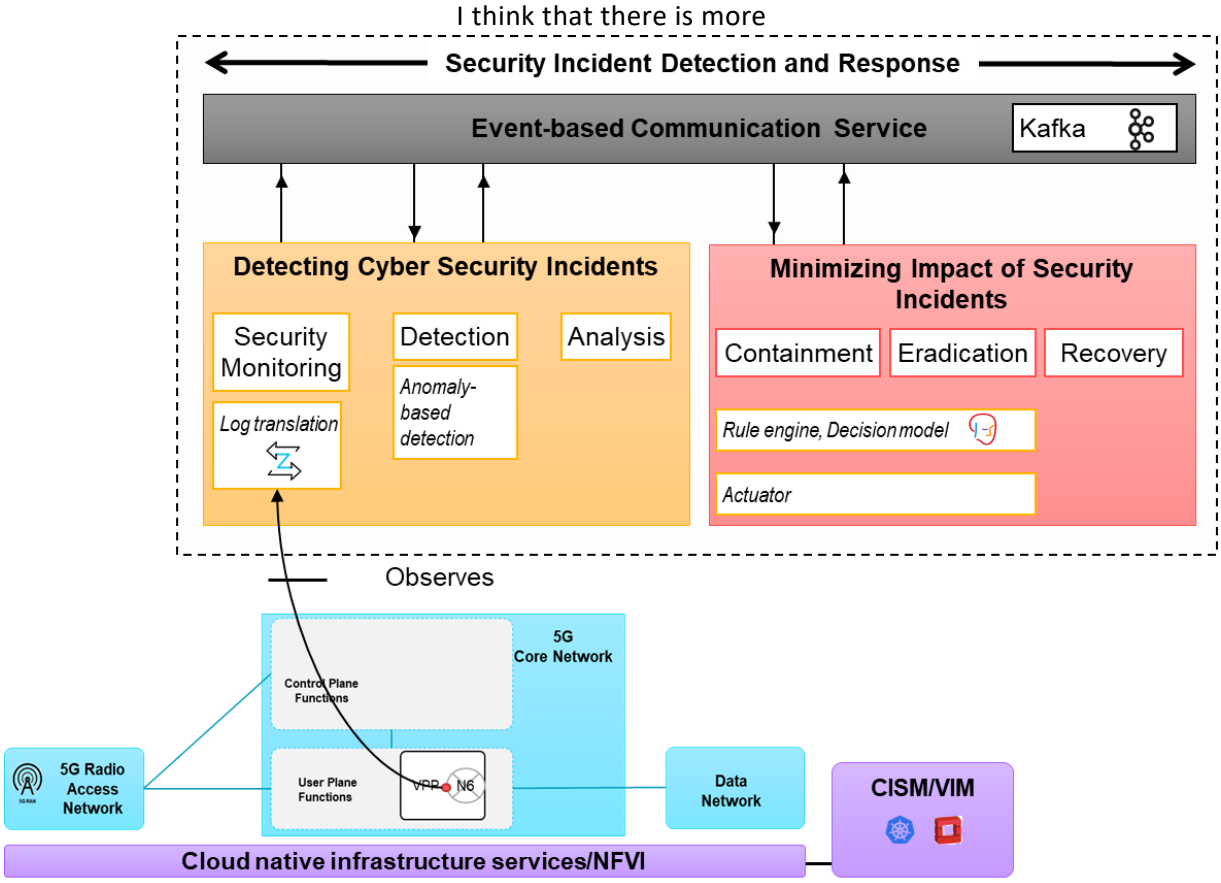


Figure 3: MS and AE components inside the global cybersecurity platform

2.1.1.2 IMPLEMENTATION OF THE REMEDIATIONS COMPONENTS

In this section, we will describe in detail the implementation of the MonB5G DE, which is in charge of limiting the impact of a cybersecurity incident based on the incident report raised by the core function detection. Based on the type of incident, the MonB5G DE generates a remediation action plan and requests its execution through the ACTUATOR.

The implementation of the MonB5G DE to respond to a security incident is based on the Business Rules Management System Drools[3]. Drools is an inference engine and a software package corresponding to a simulation algorithm for deductive reasoning. It allows expert systems to conduct logical reasoning and derive conclusions from a facts base and a knowledge base. The Decision Engine works using the following components shown in Figure 4:

- Rules: They are the reasoning part of knowledge, they are expressed in the form of logical expressions “When specific conditions occur, then do some tasks”; Once the "when" part is satisfied, the "then" part is triggered.
- Facts: external events that enter the DE. The inference engine of the DE applies the rules conditions to the event to derive new facts or actions ;
- Production memory: location where rules are stored in the Decision Engine;
- Working memory: location where facts are stored in the Decision Engine. During the runtime, these objects can be inserted, updated, deleted from the working memory by the system;
- Pattern matcher: component in charge of comparing the facts with the rules in the production memory. When a rule is fully matched, a rule match is created, referencing the rule and matching facts, and placed in the agenda;
- Agenda: component in charge of sorting the rules that have matched for a fact and that may be in conflict. It takes into account priorities in case a same fact triggers several rules and, thus, creates the adequate execution order for the rules.

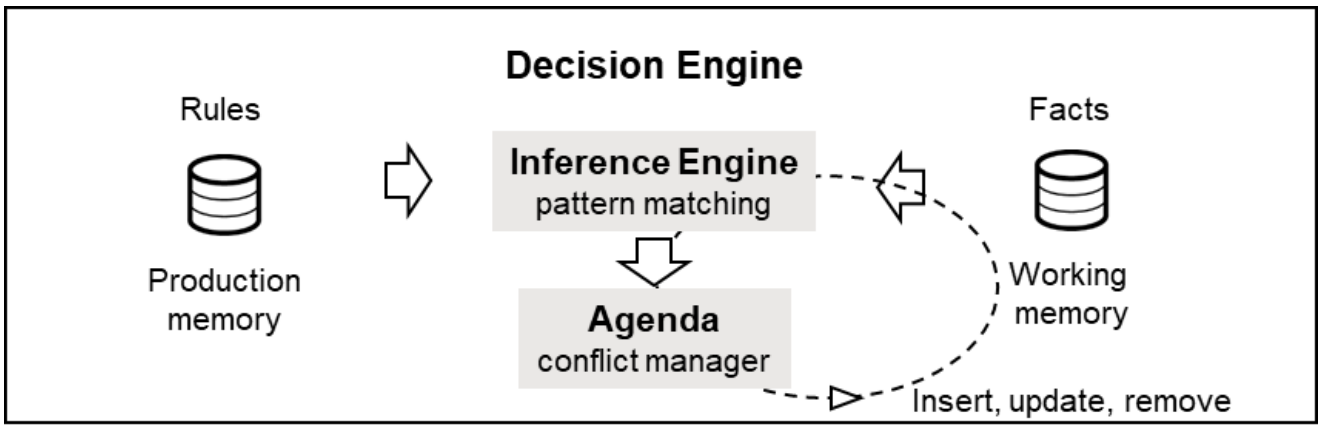


Figure 4: Abstraction for the Decision Engine

[3] https://www.drools.org/

Drools Workbench is used, for the development of our rules and business processes. It contains a graphical interface to deploy rules and business processes in the rule execution server called KIE using interfaces such as REST or JMS, which facilitates implementation.

On the one hand, the DE is a Kafka consumer, which subscribes to the desired topic and receives messages in the format it can handle. Relevant data is extracted from the message and inserted into the working memory, then the inference engine triggers the rules to evaluate the inserted data and perform actions. By inferring rules about the new fact, an action plan can be generated that describes the activities required to remedy the attack. In contrast to the previous step, the MonB5G DE will produce events containing the action plan and write them in JSON format to a dedicated Kafka topic. Therefore, it is necessary at first to configure Kafka with the appropriate topics used by the DE.

For aLTEr attack, the input of the Decision Engine will be an Event object in JSON format with the following fields:

- domain_name (string): domain name of the DNS request, such as "https://google.com";
- event (string): type of the attack, here "aLTEr_attack";
- src_ip (string): IP address of the UE trying to connect to the domain name

As an example of input event:

- "ndr.ae.alert.alter" input topic for aLTEr process. There, messages will be similar to: {"data":{"event":"aLTEr_attack", "domain_name":"www.google.com", "src_ip" :"10.51.111.180"}}

. As for the action plans, there is only one output topic, "ndr.drools.actionplans". This corresponding output of the MonB5G DE will be an ActionPlans object including the following fields:

- action_plan_name (string): type of remediation to be performed at the actuator level, here "dot";
- description (string): description of the remediation to be performed at the actuator level, here "dns over tls action plan";
- src_ip (string): IP address of the UE trying to connect to the domain name.

As an example of output event:

- for aLTEr process: {"action_plan_name": "dot", "description": "dns over tls action plan", " src_ip": "10.51.111.180"}

The ACT is a Python program that translates action plans received from the MonB5G DE into API calls. The action plan contains the parameters of the elements involved in the attack, and the desired outcome, so the ACT can translate this into atomic actions. For example, atomic actions consist of triggering the API to:

- Manage the configuration of a security function; for instance, add or deny a traffic flow in the Firewall (FW).
- Manage the life cycle of a service; for instance, add or remove DoT (DNS over TLS) in front of the DNS server. To receive the action plans from the DE, the actuator consumes action plans from a dedicated topic in the event-based communication service.

As for the aLTEr scenario, the attacker exploits the fact that data integrity is optional at Data Plane level to manipulate users' DNS requests, in order to redirect those requests to a malicious DNS server. This

manipulation occurs over the radio link, when the request can be intercepted, as represented in Figure 5. As shown in this figure, the current implementation relies on a simulation tool, the UERANSIM, which simulates the UE, the RAN and the air interface. From the point of view of the rest of the system there is no difference between this setup and a real RAN and UE, for the purpose of our demonstration.

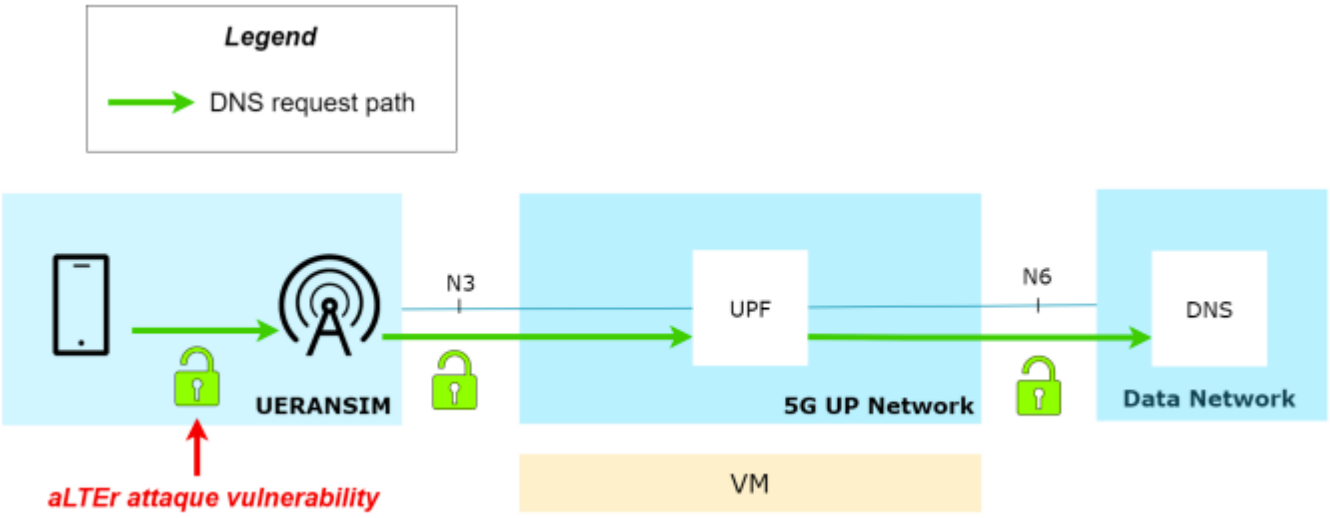


Figure 5: Vulnerability to aLTEr attack

The chosen remediation here is to activate DNS over TLS (DoT) in order to ensure the integrity, authentication and encryption of the transport layer between the UE and the DNS server. This requires to apply the appropriate settings to both UEs and the DNS server. While UE are usually commercial devices, which are expected to implement all the required protocols, the DNS server will not necessarily have this capability. To solve this issue, the remediation process includes the deployment of a CoreDNS[4] to support DoT between the DNS server and the UE. This element will act as a proxy and as an endpoint for the DoT. The remediation workflow currently implemented is represented in Figure 6. Upon reception of the remediation action plan by the actuator, a five-step process begins:

1. The actuator contacts the K8s Master Node via the master API to request the creation of a DoT proxy pod, along with a NodePort type service to make this pod reachable from outside of the cluster.
2. The requested CoreDNS pod and service are created on the K8s cluster hosting the Control Plane.
3. Then, the actuator sends a request to the Control Plane (CP) to update the UE's DNS configuration in order to use DoT as the DNS server that has just been created. This message sent by the actuator contains the following information: IP addresses of the UE and the DoT parameters, such as its domain name, port and certificate
4. The 5G core communicates with the UE via Non-Access Stratum (NAS) message to request it to use DoT with the transmitted parameters.
5. The UE updates its DNS configuration and communicates with DoT proxy, which relays plain text messages to the original DNS .

[4] https://coredns.io/

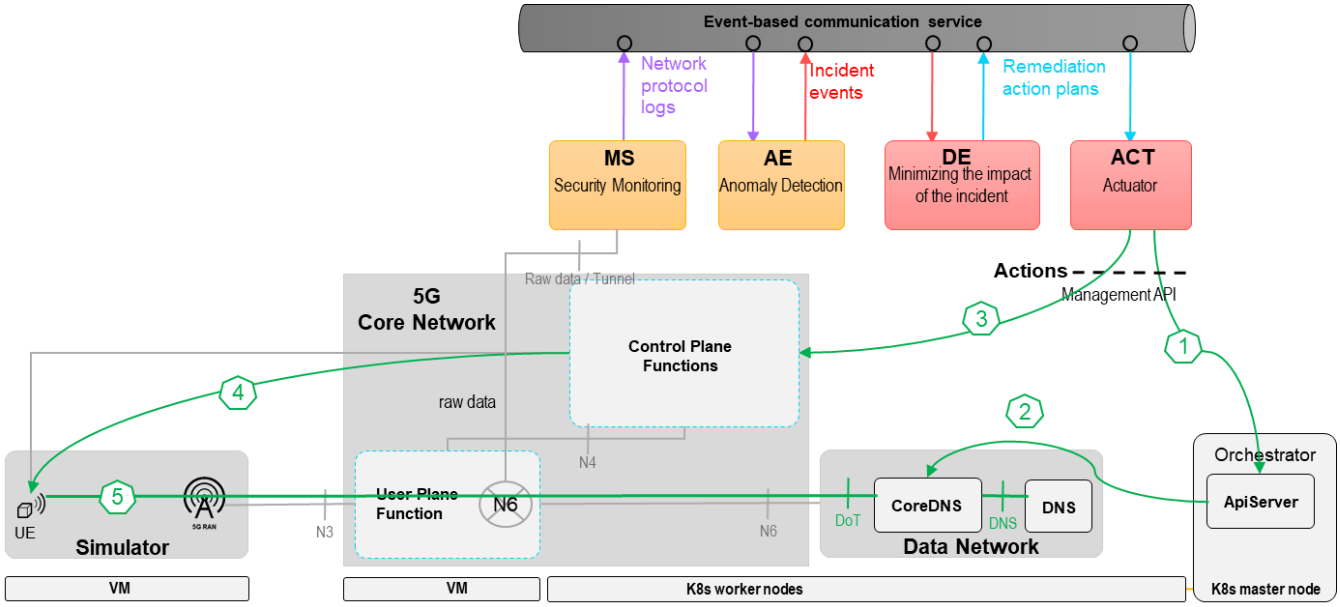


Figure 6: The projected remediation to aLTEr attack using DNS over TLS (DoT)

Currently, the 5GS CP does not yet support API to modify the network configuration of the PDU session; for the realisation of the PoC we will emulate the execution of the action of the ACT by directly updating the DNS settings on the UE.

2.1.2 SCENARIO MMTC ATTACK

To validate the proposed zero-touch security management system, we have used a 5G testbed deployed at EURECOM. The testbed has been developed and used in many 5G European projects such as 5G-EVE[5] and 5G!Drones[6]. We have implemented the closed-loop control components (i.e. MS, AE, and DE) and an Element Manager (EM) on top of the AMF. The latter exposes API to 1/ MS to monitor the Attach Request message; 2/ DE to detach and blacklist UEs involved in an attack. Figure 7 illustrates the different technologies used to implement the above-mentioned components. The roles of the different components are:

- MS and sampler: MS is the first component to receive traffic from the 5G CN. It performs basic filtering on it. While the sampler applies sampling of the input data, it receives information on Attach Requests as they are received (with no guaranteed periodicity) and emits periodic data, with the number of Attach Requests received in time intervals of a given length.

[5] https://www.5g-eve.eu/

[6] https://5gdrones.eu/

- Activity and Event Detectors: These components receive the sampled data and should detect an event. For each event, these components only emit data at its end, with the number of requests on each time-slice and all the UEs that emitted traffic during the event.
- Analysis Component: This component runs the ML algorithm on the given data, calculating a detection rate for each time-slice (for all devices in the time-slice).
- DE: This component receives data from the Analysis Component and decides which devices should be disconnected from the network and then blacklisted.
- MQTT Broker is used to implement the communication bus between the different components of the closed-loop control system, and between the closed-loop control system and the AMF.

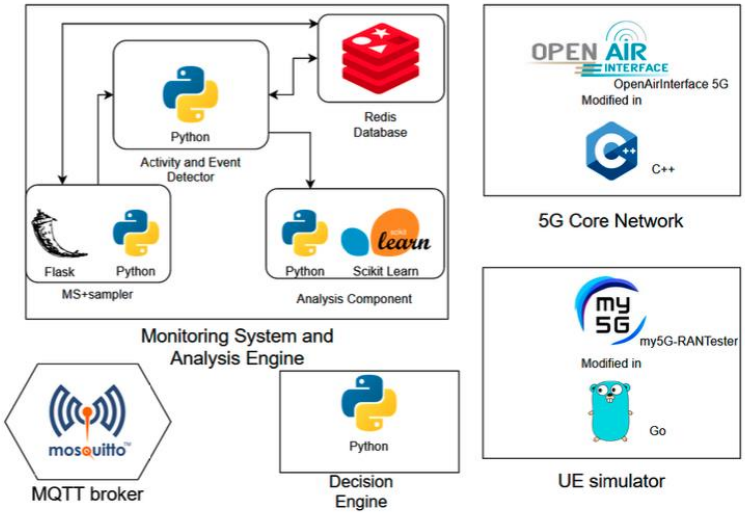


Figure 7: Test platform and technological components for mMTC attacks

2.1.2.1 IMPLEMENTATION OF THE MMTC ATTACK DETECTION COMPONENTS

MS component collects data from AMF on every Attach Request received from the MTC devices. This data is accessible through the API exposed by EM of AMF. For each Attach Request, MS extracts the device identifier Subscription Permanent Identifier (SUPI) and a precise timestamp. Indeed, in the 5G protocol, each UE is identified with a unique identifier called SUPI. The latter is encrypted to provide better privacy and prevent the IMSI catcher attacks that were popular on previous-generation telecommunication protocols. The SUPI should not be transferred in clear text over the RAN except routing information, e.g., Mobile Country Code (MCC) and Mobile Network Code (MNC). For this reason, it is challenging to identify devices from the traffic received on the radio layer; hence our solution has to intervene at the 5G CN, i.e. at AMF level, which can decrypt the SUPI information. The extracted information is then forwarded to AE component via a communication bus based on the Publish/Subscribe concept.

Regarding the UE, we used and updated a 5G UE emulator, my5G-RANTester[7], to be able to generate a high number of UE Attach Request messages in parallel to simulate an attack or normal traffic. Indeed, my5G-

[7] https://github.com/my5G/my5G-RANTester

RANTester is a tool for emulating control and data planes of the UEs and gNB. my5G-RANTester follows the 3GPP Release 15 standard for RAN. By using my5G-RANTester, it is possible to generate different workloads and test several functionalities of a 5G CN, including its compliance with the 3GPP standards. Scalability is also a relevant feature of the my5GRANTester, which can mimic the behaviour of a large number of UEs and gNBs simultaneously accessing a 5G CN. Currently, the wireless channel is not implemented in the tool. The AMF and 5G CN components are based on OAI[8].

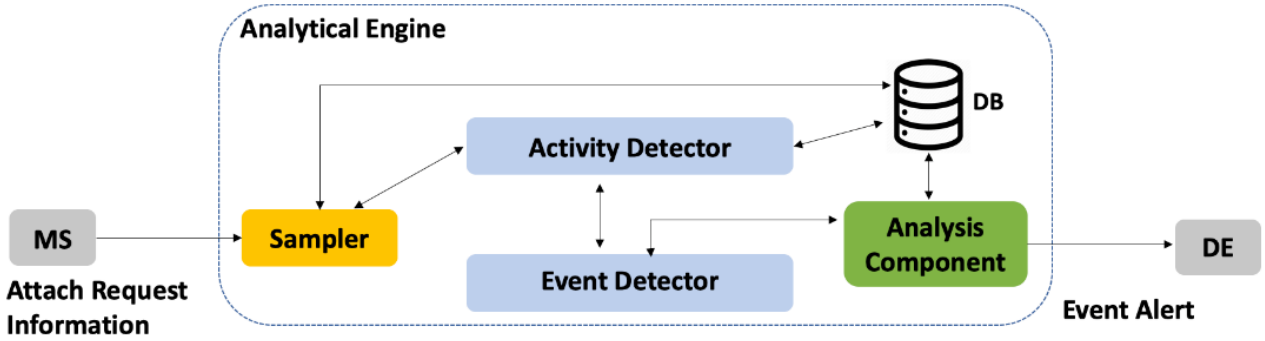


Figure 8: Analytics Engine component

Figure 8 illustrates a detailed vision of the AE components, which are: Sampler, Activity Detector, DataBase (DB), Event Detector, and Analysis components. They interact together to: 1/ detect when an event starts and ends. It can correspond to MTC devices report (normal traffic) or attacks; 2/ analyze the event to detect if it is normal or abnormal traffic; 3/ compute the detection rate for each device (probability that a device has participated in the attack) and send a report to DE. We decided to separate the event detection from event analysis to improve performances and consider all the relevant data when running the overall attack detection algorithm. Indeed, we decided to detect activity periods (i.e., events) in the network traffic and only feed data to the ML algorithm at the end of an event, which provides the advantage that the resource-intensive component (detection analysis) runs once every event. We also consider two corner cases: 1/ after a duration clearly greater than the maximum length of an event; 2/ when peak traffic exceeds a limit indicating that it is definitely an attack. For both cases, we tag the devices as malicious.

The only downside of separating event detection from the analysis is that UEs participating in an attack will not get banned right away when the attack starts. However, since the damage in DDoS attacks stems from their duration in time, the devices will get disconnected and blacklisted a few seconds or minutes after the event starts by DE. The detection algorithm does not need to run in real-time, and it can look at data of the whole event.

2.1.2.2 IMPLEMENTATION OF THE MMTC ATTACKS MITIGATION COMPONENTS

DE component is the decision-maker of the closed-loop control system. It receives data from AE and decides the actions to take for UEs that emit abnormal traffic. DE gets as input a list including the suspected UEs (SUPI) and their corresponding detection rate values belonging to the attacks. We devise two versions of DE.

[8] https://openairinterface.org/

The first solution blacklists devices if their calculated prediction is higher than "DE DETECTION THRESHOLD" (i.e, a configurable parameter). The lower the threshold value, the higher the probability that devices are blacklisted. Therefore, the network operator would use lower values in order to be stricter, but in turn, it may increase the false positive. To avoid having high false-positive results, we introduce a second solution that relies on three thresholds. This solution considers the whole event and classifies the received list of UEs into three categories: F_1, F_2 and F_3 (Figure 9). DE calculates how many UEs have obtained a detection rate (detection_i) higher than 0.8 and assigns it to the first category, namely F_1. The second category includes UE having a detection rate between 0.3 and 0.8. This category corresponds to F_2. The remaining UEs are included in F_3. Then, DE checks if, in the event, a significant part of the devices had higher than usual detection rates. As a result, different decisions are to be taken:

- First, if F_1 > F_2 and F_1 > F_3, DE blacklists all the devices by adding their SUPI values to a table of blacklisted values, and disconnect them from the network.
- Second, if F_2 > F_1 and F_2 > F_3, DE adds the SUPI values to a table named ``non-trusted devices". Each UE belonging to this table has a counter named T_imsi. The counter is increased by 1 each time the UE is involved in abnormal traffic that is not blatantly an attack. The counter is incremented until it reaches a value that yields to blacklist the device.
- Third, DE ignores the alert sent by AE, and it will do nothing.

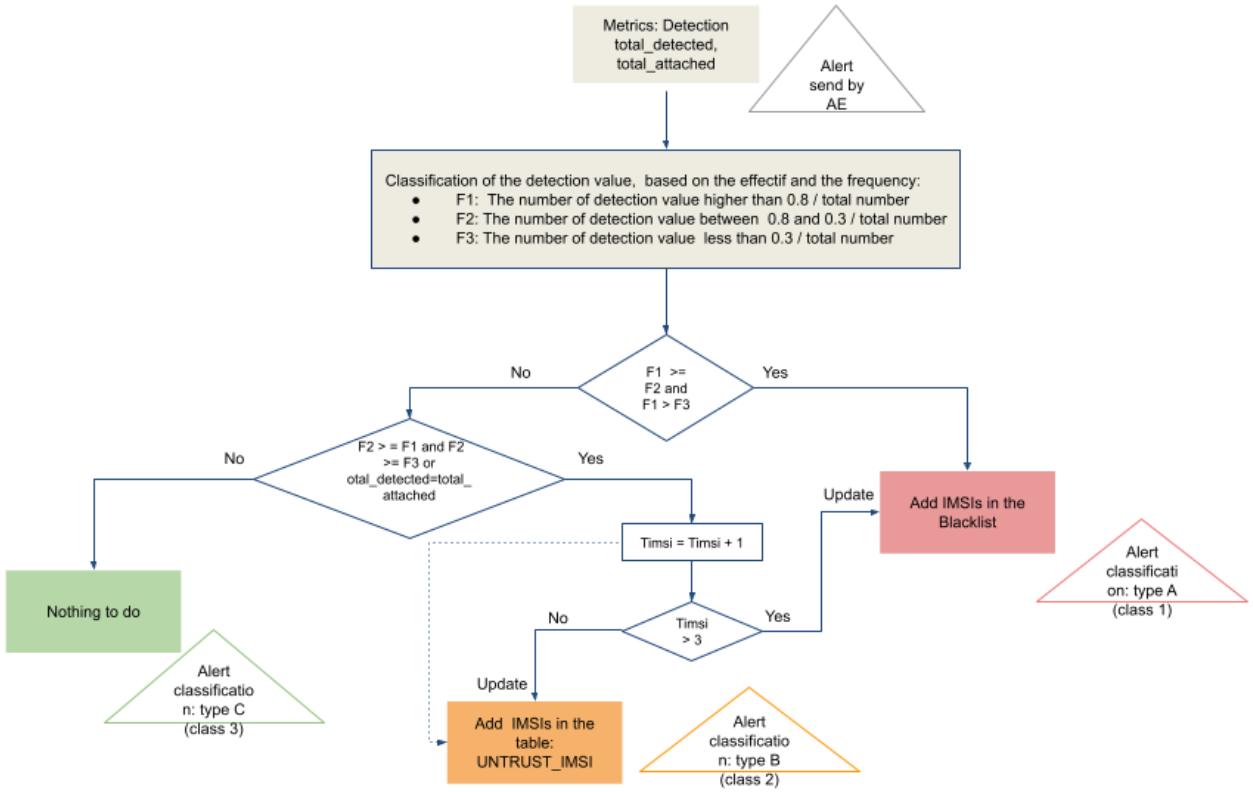


Figure 9: Flowchart of the DE's components

The reason behind using 3 categories is based on the AE analysis and the results accuracy of the employed ML algorithm. Indeed, we notice that when the detection values associated with the connected devices sent by AE are high, the devices' generated traffic does not follow the β Beta distribution. Hence, they should be immediately blacklisted as they present abnormal behaviour, justifying the need for the first category. The latter is considered a red alert, and the associated devices SUPI are blacklisted. However, when the values are neither too high nor too low, it means that the traffic is almost close to the Beta distribution. In this case, the detected devices have malicious behaviour, or the values are due to technical failure. So, we introduced the second category, which means that the devices are not blacklisted, but DE will memorize the associated SUPI for future events. If the devices are classified in the second category more than 2 times, they will be considered malicious and moved to the first category to be blacklisted. The last category corresponds to the detected traffic being very close to the Beta distribution. This case can be either an error in the ML calculation or a delay in sending the Attach Request.

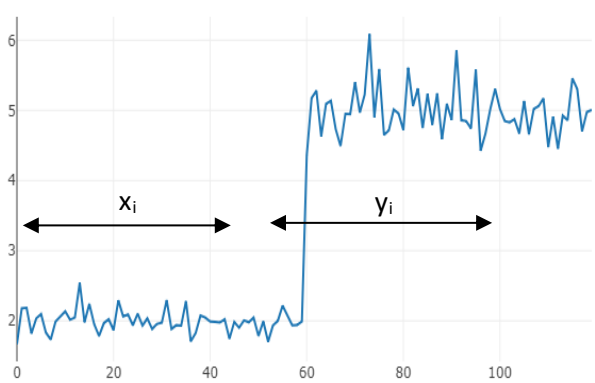
2.1.2.3 ALTERNATIVE IMPLEMENTATION OF THE MMTC ATTACK DETECTION COMPONENTS

In order to assess the flexibility of the MonB5G framework in supporting various different algorithms, we implemented an alternative detection algorithm for the mMTC scenario.

The detection algorithm is a variation and (our own) Python-based implementation of Twitter's Breakout Detection[9], which detects behavioural change in time series data. The underlying algorithm – referred to in [JAM16] as E-Divisive with Medians (EDM) – employs *energy statistics* to detect divergence in mean. EDM can also be used to detect change in distribution in a given time series. EDM uses robust statistical metrics and estimates the statistical significance of a breakout through a permutation test. In addition, EDM is non-parametric. This is crucial, since the distribution of production data only seldomly, if at all, follows the commonly assumed normal distribution. The algorithm can also be used to detect multiple breakouts in a given time series.

Our implementation consists at a high-level of two stages:

1) Detecting changes in distributions, using Energy Distance.



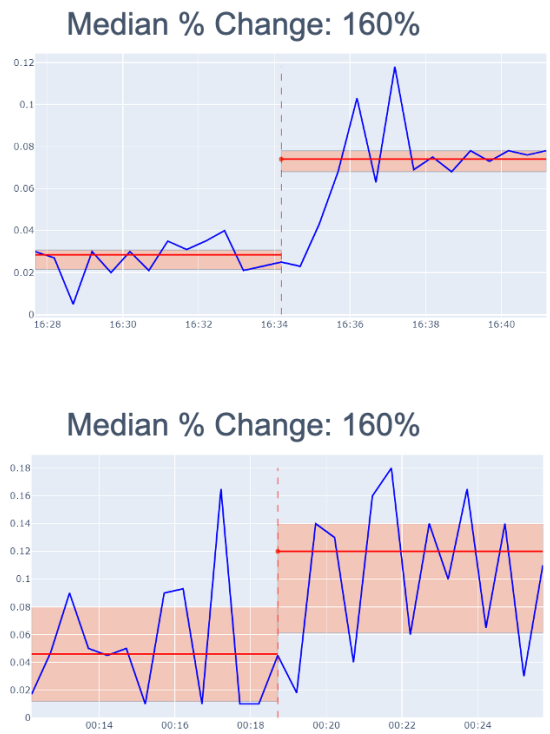
$$E(X_n,Y_m;\alpha) = 2m^{\alpha}_{XY} - m^{\alpha}_{XX} - m^{\alpha}_{YY}$$
$$m^{\alpha}_{XY} = \text{median}\ \{|x_i - y_j|^{\alpha} : 1 \le i \le n,\ 1 \le j \le m\}$$
$$m^{\alpha}_{XX} = \text{median}\ \{|x_i - x_j|^{\alpha} : 1 \le i < j \le n\}$$
$$m^{\alpha}_{YY} = \text{median}\ \{|y_i - y_j|^{\alpha} : 1 \le i < j \le m\}$$

Figure 10: Energy Distance calculation

[9] https://github.com/twitter/BreakoutDetection

Breakout detection uses the *energy distance* E between two sets of samples as a measure of similarity. The higher the value of this statistic, the higher is the difference between the two sets. The idea is based on calculating the energy distance between two random variables. The random variables X and Y correspond to the two set of samples that are being examined for a potential breakout. To make the E metric robust to anomalies of the signal (such as spikes or dips), the authors replaced the means in the original formula with medians, as shown in the formula above. The distance is parametrized by a (0<a≤2) and the authors state that using a=2 will result in detecting divergence in the mean, while a setting of (0<a<2) can be used for detecting arbitrary changes in the distribution (our implementation apparently uses a=2). The m^{α}_{XY} in the formula refers to the between distance of the two set of samples, while m^{α}_{XX} and m^{α}_{YY} refer to the within distance of X and Y respectively.

2) Filtering out breakouts not worthy of detection, using Bootstrapping.



Confidence Interval (95%) for the median % change of the two sets

95% CI = [120, 240]

The lower bound of the CIs captures the difference between the two sets

95% CI = [0.2, 90]

Figure 11: Assessing significance of detected breakouts using confidence intervals

While evaluating the results of breakout detection in our testbed, we noticed a higher than desired detection rate, that was dominated by the presence of "borderline" cases, such as the second example of the two examples above. For that reason, we introduced a filtering stage that excludes the borderline detections and increases the true positive rate, sacrificing some false negative cases. By introducing a statistical test for the filtering procedure, it is possible to construct a X% confidence interval, within which we can claim that with X% confidence lies the true median (our implementation uses 95%). Bootstrapping[10] is such a non-parametric statistical test, which essentially samples (with replacement) a population many times, computes a statistic

[10] https://en.wikipedia.org/wiki/Bootstrapping_(statistics)

and then constructs a confidence interval (CI) based on all of the statistics for all of the sampled populations. By implementing this filtering technique, we were able to distinguish between the two classes of desirable vs undesirable detections.

Over the course of the project, we have developed and tested two implementations of the above algorithm:

1. Batch, appropriate for executing in Jupyter notebooks or batch apps. This has minimum set of dependencies: Python, numpy, pandas, breakout_detection, itertools, etc.

2. Streaming, better suited for PoC implementation. This has more complex dependencies: Docker (container), Kafka, Python, Faust, plus the dependencies outlined above.

The streaming implementation, at its core, must detect breakouts in the Attach Request message rate generated by the AMFs of the mMTC use case. During the operation of the application, enough data from each AMF and slice is retrieved from the Input and buffered internally, in order to perform the breakout analysis. If a breakout is detected, all data for this occurrence needs to be discarded and the analysis should continue after enough "new" data is collected. This is also referred below as the skipping strategy of the application. Finally, if a breakout event is detected, the application reports it to an Output sink.

The Streaming Application implements the above via a series of asynchronous processing agents and respective Kafka topics for serialization of the whole process. Each of the agents performs a distinct function of the application. These functions are triggered by events in the input Kafka topic of each agent. There are three agents in this application: 1/ the Stream Prepare ("PreProcess") agent, 2/ the EventState Manager ("State Update") agent and 3/ the State Process agent. In order to bind each agent with events of a Kafka topic, three topics are needed, respectively, and one additional for posting the output of the application.

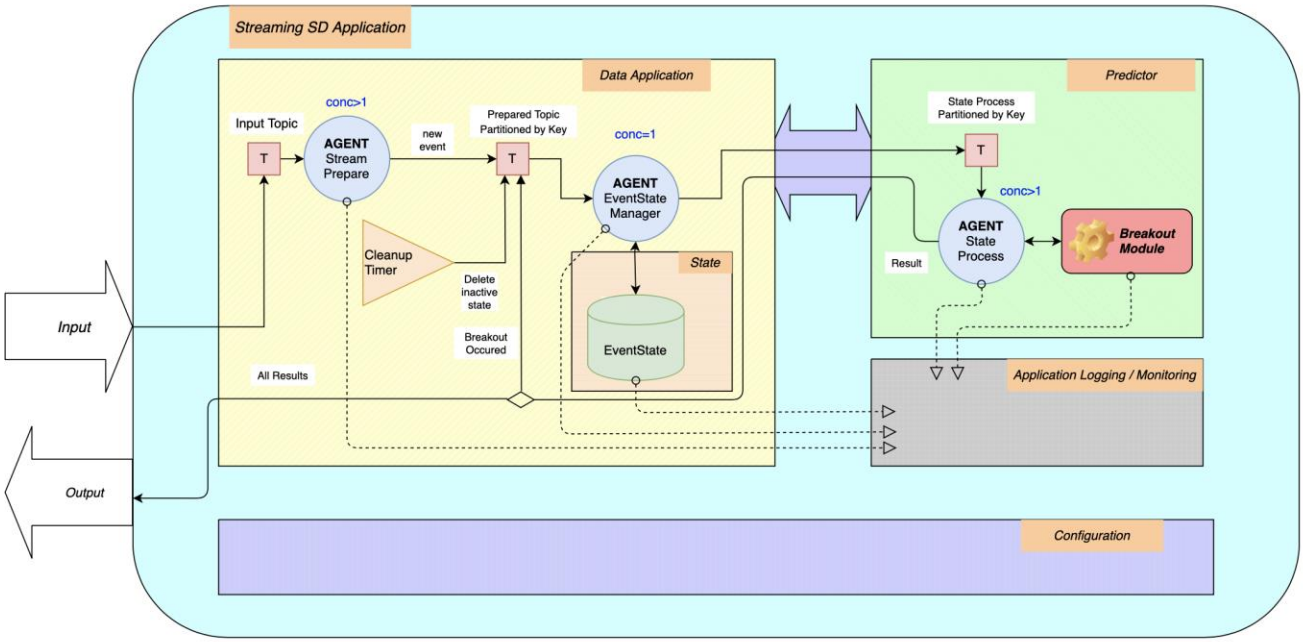


Figure 12: Block diagram of Analytics Engine implementation

The above components are visually presented in Figure 12. Each component and its operation are analyzed in more detail below:

- The PreProcess agent reads input messages from the Input topic, which carries data from multiple AMFs and slices, but also data potentially redundant for the purposes of this application. It then filters and transforms each input message to multiple processed messages, separately per slice. These messages then contain only data relevant to this AE. The PreProcess agent posts each message to the "Prepared Topic" Kafka topic.
- The State Update agent is responsible for updating the state of the application. This agent reads messages from the Prepared Topic and updates the state of the application. If the state of the application has sufficient data for the breakout detection module to be executed, the State Update agent posts the data of the state towards the State Process agent as a timeseries. Note, that these timeseries are separated per slice. The Prepared Topic contains two more types of messages, one for cleanup, if needed, and one that indicates that a breakout has happened. The cleanup timer triggers an action that cleans old messages from the state, while the breakout messages facilitate removal of duplicate breakouts detections and trigger the skipping strategy of the application.
- The timeseries per slice posted at the State Process topic are consumed by the State Process agent, where the breakout module detects if there is a breakout occurring. The result of the breakout module is the output of the application. Note that, if a breakout is detected, the result is also sent to the State Update agent, where it is registered at the state, so that the skipping strategy can be implemented.
- The Breakout Module is the core of the application. In this module, the timeseries per slice processed in the other parts of the application are analyzed to determine if a breakout occurred. The general logic is that each timeseries is divided in two sub-series using sliding windows, the first and second half, and the two corresponding distributions are examined for behavioural changes. First the data is sorted in time order and the values of the timeseries are computed. Then the data are checked for validity and are normalized. If the resulting timeseries fulfils admission criteria, it is tested through the breakout detection library, results are verified via a permutation test and filtered through a median percentage change test (using the bootstrapping algorithm), before reporting a breakout in the output. If any of these tests fail, no breakout is reported.

2.1.3 SCENARIO FEDERATED LEARNING ATTACK

As depicted in Figure 13, we consider n running network slices that may be initiated by different vertical industries, such as intelligent transportation, Industrial IoT, and eHealth verticals. The running network slices are interconnected to an Inter-Domain Slice Manager (IDSM), which is in charge for the management and orchestration of network slices. To enable ZSM, the IDSM side includes an AE for building learning models and a DE, to make suitable decisions based on AE's outputs. On the other side, each running network slice is managed locally by a Domain Slice Manager (DSM), which also includes a MS for monitoring data and in-slice traffic, and an AE for building learning models.

The proposed framework enables to secure federated learning in B5G networks, against poisoning attacks, named TQFL for "Trust deep Q-learning Federated Learning". The design of our framework comprises four main steps, starting from generating a realistic dataset to designing a detection scheme of poisoning attacks: (i) The generation of a realistic dataset about the AMF function's latency of running network slices and its

(latency) related parameters. (ii) Building a deep learning model to predict the AMF function's latency of each running network slice in a federated way in order to prevent any latency-related SLA violation. (iii) Building an online Deep Reinforcement Learning (DRL) model that dynamically selects a network slice as a trusted participant (see step 1). (iv) After the first FL rounds (see steps 2, 3), the trusted participant applies a dimensionality reduction scheme and unsupervised machine/deep learning to detect the malicious participant (s) (see steps 4, 5, 6, 7).

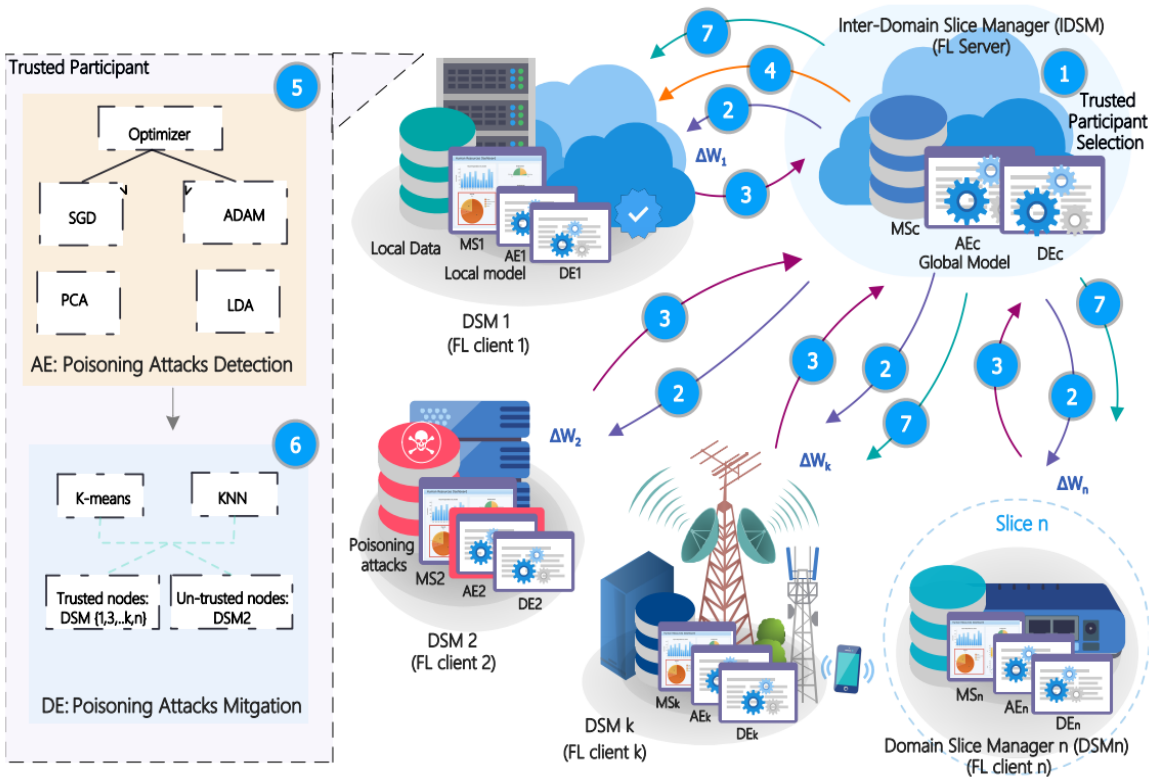


Figure 13: Overview of TQFL Framework.

2.1.3.1 IMPLEMENTATION OF THE FL ATTACK DETECTION COMPONENTS

Because of the lack of real dataset, a real testbed using Eurecom OAI platform is setup in order to collect and generate a realistic dataset, called EARCD for Eurecom AMF Resource Consumption Dataset. OAI implements 5G radio access and core networks, as open-source software. Ten instances of AMF are emulated, running as VNFs inside ten isolated network slices. The network slices differ from each other, in terms of their AMFs' configurations, for instance: the AMF of network slice 1 has 1GB of memory and 1 CPU, the AMF of network slice 2 has 2GB of memory and 2 CPUs, etc.

In addition, my5G-RANTester tool is used to emulate user equipments (UEs), one gNB and to generate attach requests, that will be then handled by the different network slices' AMFs. By increasing the number of Ues, up to 560 attach request per second could be generated, covering different traffic density.

Therefore, ten local datasets (ten network slices) were generated, by varying the number of handled attach request/s, while each dataset contains 2813 samples (rows). In addition, each local dataset contains five features as input data, including RAM capacity, CPU capacity, RAM used, CPU used, and number of attach request, and latency in terms of average duration of Ues attachment, as output data. The latter corresponds to the response time (latency in second), to handle UE attach requests by the network slices' AMFs.

This section presents the poisoning attack detection scheme, which consists first to elect a network slice participant as a trusted entity. The latter will then be in charge of detecting malicious clients, by leveraging unsupervised learning (Dimensionality Reduction algorithms).

a. Trusted Participant Selection using Deep Reinforcement learning

At each FL round, the central IDSM of running network slices selects a running network slice (DSM), as trust node. To do so, we design a new deep reinforcement learning-based model, to derive an optimal policy about trust node selection, while considering several criteria related to such nodes, such as their reputation, detection rate of malicious nodes, and its accuracy in building learning models.

Deep Reinforcement learning is a process that enables one or set of agents to learn on how to make suitable decisions, through error and trial, and based on their (agents) previous experiences. Specifically, each agent interacts with the environment to receive either penalties or rewards, for the actions it made. Hence, the main objective of deep reinforcement learning is to derive an optimal policy about agents' actions that maximizes agents' cumulative reward. In our study, the central IDSM is the agent that interacts with running network slices' DSM (environment), in discrete time steps, as shown in Figure 13.

b. Dimensionality Reduction of model updates

Once a trust client is selected, it will be in charge of detecting whether the received updates include a malicious model or not. First of all, the selected trust client receives the n model updates of network slices from the central IDSM, and then applies a dimensionality reduction techniques to be able in presenting the model updates in 2D dimensions. Indeed, the dimension reduction is the transformation of dataset from a high-dimensional space into a low-dimensional space, in such a way, the low-dimensional representation retains the meaningful properties of the original data.

Figure 14 shows a dimensionality reduction of our FL model updates in 3D, when applying linear discriminant analysis reduction technique. The different colour of the points, which are defined by 3 axes (x:0, y:1, z:2), present the updates of different nodes (10 clients).

However, as we can notice, the reduction to 3D will not provide a clear visualization, to interrupt and classify the model updates. That is why we decided to apply dimensionality reduction to 2dimensions (2D) (Figure 15). In our study, to design an efficient detection scheme, we are based on two different techniques: PCA as unsupervised technique (ignores class labels), and LDA as a supervised technique.

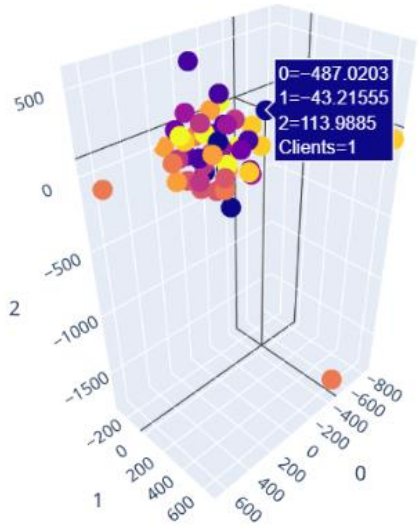


Figure 14: LDA visualization with 3 dimensional applied on the nodes updates

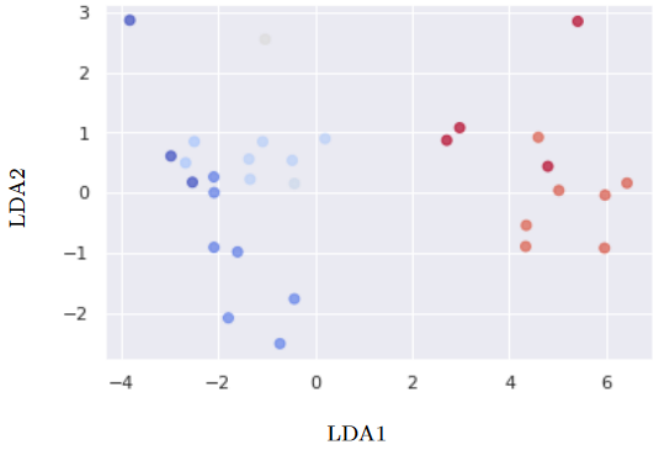


Figure 15: LDA visualization with 2 dimensional applied on the nodes updates

2.1.3.2 IMPLEMENTATION OF THE FL ATTACKS MITIGATION COMPONENTS

Once applying dimensionality reduction techniques and depicting network slices' updates in a 2D plan, the last step consists of grouping the received updates into several clusters, in order to determine malicious updates/models.

To ensure an effective detection of malicious updates, two different clustering algorithms (unsupervised and supervised) are selected: 1/ k-means which is an unsupervised learning algorithm, that considers no labelled update models' data, and 2/ k-nearest neighbours (KNN), as supervised learning algorithm which considers labelled data about model updates. Both algorithms aim to divide the received update models, at each FL

round, into k clusters that share similarities and are dissimilar to the model updates belonging to another cluster. We note that we leverage the model update of the trust participant as a reference that supports to make the difference between malicious and trust models.

2.2 Solutions Targeting Energy Efficiency

2.2.1 IMPLEMENTATION OF THE STOCHASTIC FEDERATED LEARNING COMPONENTS

The MonB5G AE analyses the performance of network slices using telemetry data collected by the MS. The results of this analysis are used by DE to make decisions to optimize slice performance and meet SLAs. The KPIs that are tracked for each slice include things like throughput, delay, and resource utilization. The number of slices and the amount of data being processed can impact the efficiency of the AE, so it has been designed to be scalable. This chapter describes prediction methods used in the AE, which has been implemented as a Docker container so that they can be deployed in cloud native environments of WP6. To ensure scalability, several strategies were employed in the design and implementation of the AE.

The main objective of the federated learning approach implemented in this solution is to test the feasibility and scalability of our proposed stochastic federated learning algorithm in a real-world scenario using cloud-native FL agents from an energy-efficiency perspective. We will use the Docker Compose tool to emulate cloud deployments, as it typically runs on top of Kubernetes and is supported by Container Orchestration Engines (COE) offered as a Platform as a Service (PaaS).

In the cloud-native implementation of FL agents shown in Figure 16, the FL Server is contained within a single Docker container. Meanwhile, multiple AE clients run simultaneously.

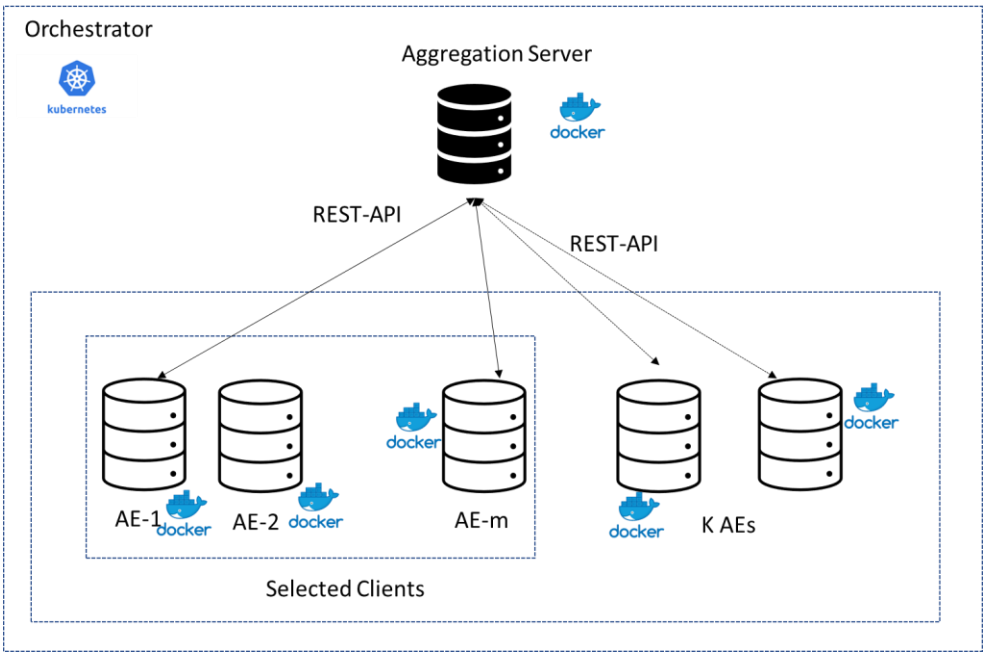


Figure 16: Implementation of a scalable agent selection system using cloud-native technology

The communication process in this system involves two main modules: a server module that conducts training among clients and a module that is used by all clients participating in the FL training process. These modules communicate with each other using the HTTP protocol and various REST interfaces that are given in tables below. Table 1 Server Side (4 APIs)Server Side (4 APIs)shows 4 APIs that are available at Aggregation Server so that Admin and AEs/clients can communicate through.

APIs	Description
POST/client	Registering clients with the Server (from Client to Server)
GET/select clients	Initiate policy for selecting clients and corresponding FL training (from Admin to Server)
POST/SLA	Clients send their SLA violation rate to the Server node (from Client to Server)
PUT/model-weights	Clients send calculated model parameters to the Server node (from Client to Server

Table 1: Server Side (4 APIs)

Table 2 shows 3 APIs that are available at AEs so that Aggregation Server can communicate through.

APIs	Description
PUT/SLA	Server requests each of the clients to calculate their SLA violation rate (from Server to Client)
POST/training	Server requests the selected clients to start FL training with new model weights (from Server to Client)
PUT/worker_model	Update client initial model parameters. (from Server to Client)

Table 2: Client Side (3 APIs)

Procedure for the operation of the entire network

Figure 17 shows the Swimlane flowchart of the cloud native implementation of the proposed stochastic FL approach. To run the network properly, the system needs the server node to be running and at least one client node available for training. Once the client nodes are aware of the IP address of the server, they can register with the server using a POST/client request with their own IP address. The server will then send requests to all registered clients to start the proposed client selection policy using a POST/select-client request as in step-2. The clients will calculate and send their associated SLA violation rate to the server using

PUT/SLA and POST/SLA requests in step-3. The server will then generate a probability distribution of the clients using the softmin function and select clients for training using the np.random.choice function in step-4. The server will send POST/training requests to the selected clients to start FL training in step-5. After local training is complete in step-6, the clients will send their model weights to the server using PUT/model-weights in step-7, and the server will calculate the average of the model weights in step-8 and update the overall system in step-9. This process will be repeated for each FL round to step 3 if no convergence occurs.

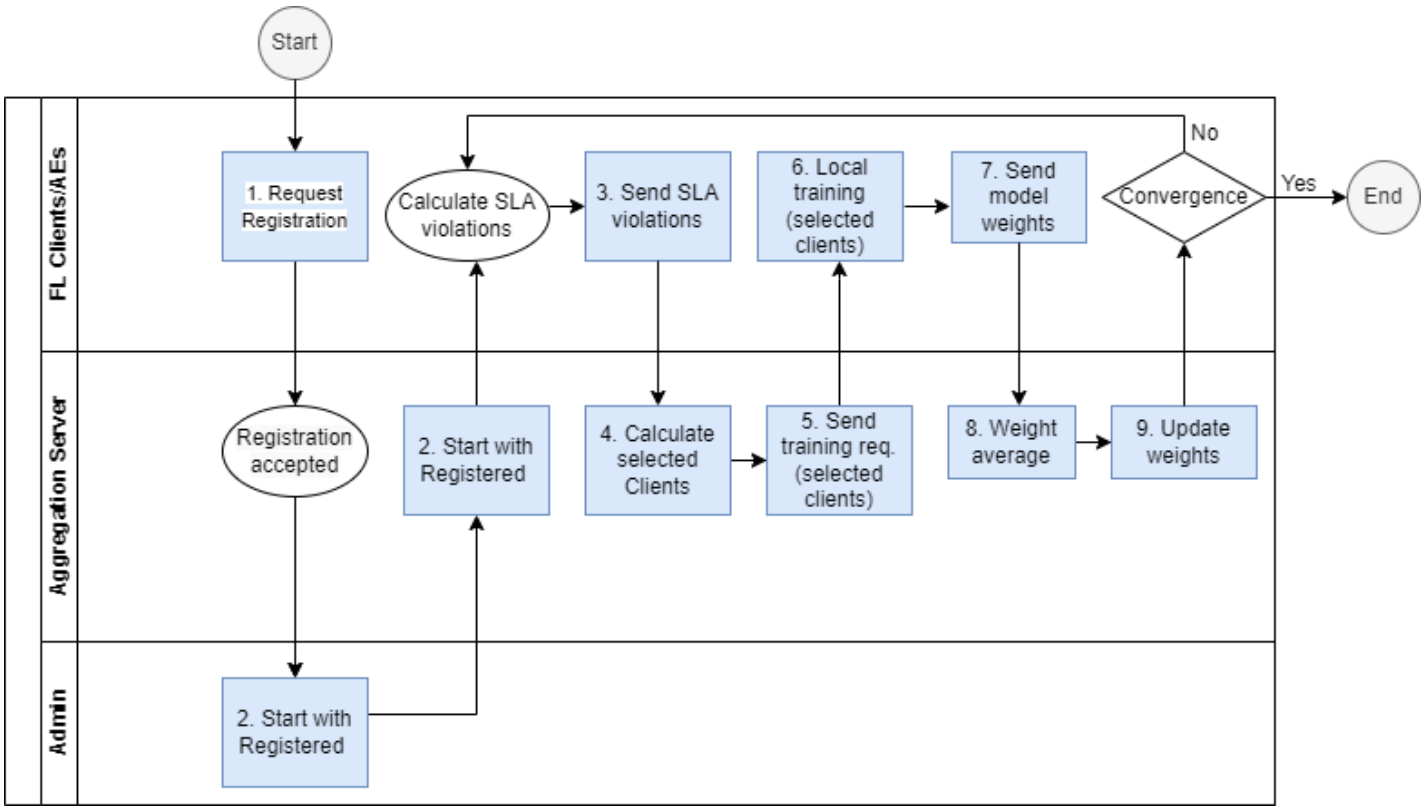


Figure 17: Swimlane flowchart of the cloud native implementation of stochastic FL approach

2.2.2 MULTI-AGENT, ENERGY EFFICIENT VNF ORCHESTRATION

The work is an innovative, multi-agent, energy-aware, RL-based, VNF orchestration framework for multi-domain B5G networks. The proposed DE and ACT are distributed, eliminating any central point of failure and enabling scalability.

The proposed framework enables energy-efficient VNF orchestration in multi-domain networks by installing multiple DEs and ACTs in each individual network domain, for each network slice. Multiple RL agents are instantiated in each domain and perform VNF orchestration locally. A system for inter-domain migration assists the local DEs by scheduling and performing VNF migration between the domains when the efficiency KPIs are not expected to be met by the projected network conditions.

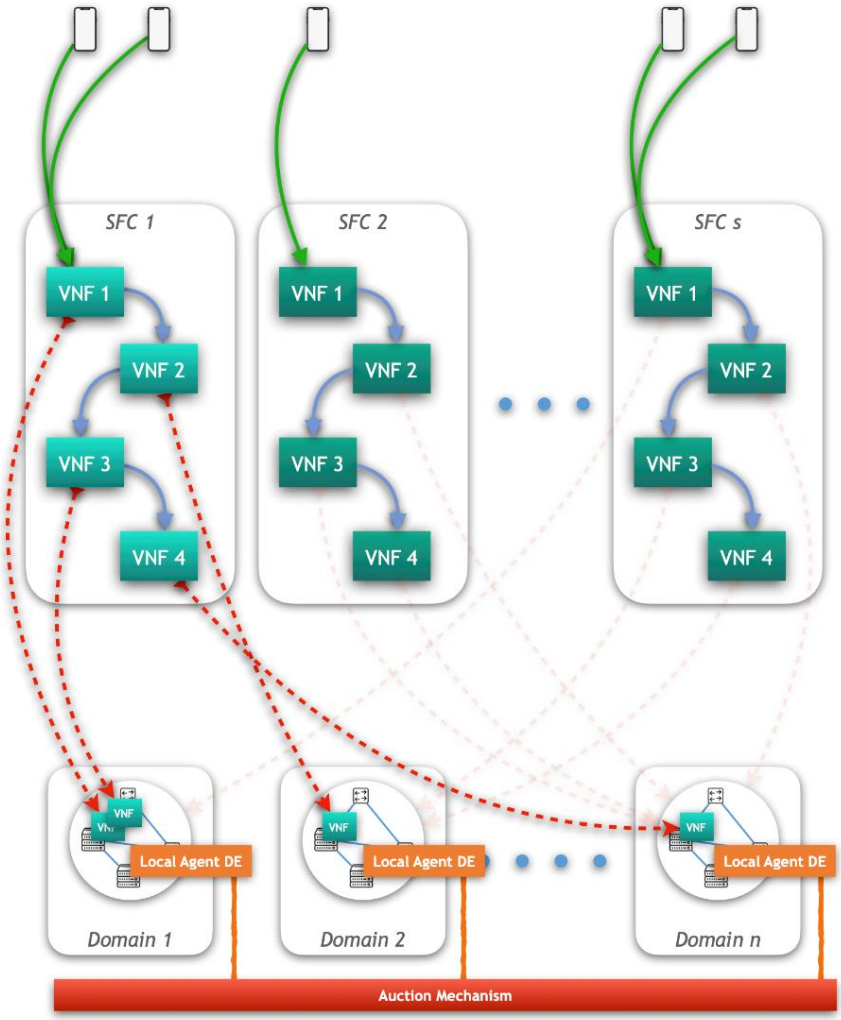


Figure 18: Inter-domain VNF chain migration framework in a network slice.

There are multiple remediations components in the framework that act to bring the system in an equilibrium between performance and the set KPIs:

- **Distributed Local Agents:** RL-based intelligent agents instantiated in each network domain, capable of orchestrating the VNF chains locally, enabling local reconfigurations and avoiding costly migrations, both in terms of cost, network stress and performance. The VNF placement problem is defined as a Markov Decision Process (MDP).
- **Auction Mechanism:** A system that enables the inter-domain VNF migration and local agent communication in a distributed multi-domain network setting. The Auction Mechanism coordinates all distributed agents via an auction-procedure that decides the VNF placement in the extended network, so the overall slice KPIs can be met.

3. Evaluation

Goal here is to measure metrics of the components developed within the scope of our scenarios.

3.1 Security

3.1.1 SCENARIO ALTER ATTACK

The scope of aLTEr attacks scenario is to display the functionalities of the Service Orchestrator (SO) and provide a comprehensive description of the MS, AE, and DE for detecting and mitigating attacks. Specifically, the aLTEr attack is an MITM attack type and is carried out between the user equipment (UE) and the gNodeB (gNB). It involves a breaking layer two of the user radio bearer, exploiting the fact that the user data integrity protection can be missing as a vulnerability to carry out the attack. To detect the attacks, AI/ML algorithms were explored to detect the aLTEr attack, executed at AE. The mitigation step executed by DE consists of a security policy update on the firewall to deny the private DNS address from the UE.

3.1.1.1 MEASUREMENT METHODOLOGY

The environment that has been set up for the testing is based either on simulation or virtualized infrastructure. When the virtualized infrastructure is used, the hosts are virtualized machines (VM) managed by OpenStack, on which we deploy applications or worker nodes of Kubernetes (K8S) cluster. The network service of the control plane of the 5G system and the MonB5G components MS, AE, DE, and ACT are K8s services and pods, while the remaining parts of the 5GS such as the data plane and the simulator of UE and RAN are applications running on the VMs. Moreover, we also keep all host clocks synchronized by the Network Time Protocol (NTP) to have consistent timestamps in logs and to correlate events.

The algorithms implemented for the anomaly detection are tested as a component of MonB5G AE deployed on the platform, or by simulation. However, the same data set is leveraged for machine learning of normal behaviours.

As for the running tests and performance measurement, the following tools are used:

- the command line tool “dig”[11] is used for querying the Domain Name System (DNS)
- Wireshark to capture raw IP data on the network interface
- Time-stamped application logs from Kafka, K8S pods, and Container Network Interface (CNI) plugin
- Microsoft Excel to process comma-separated values exported from logs and Wireshark capture files

3.1.1.2 KPI DESCRIPTION AND RESULTS

Description of models

A brief description of the implemented models and the evaluation scores for each model is described in the following section.

1. **Random Forest**

[11] https://en.wikipedia.org/wiki/Dig_(command)

Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

2. Gradient Boosting Classifier

Gradient boosting involves three elements:

1) A Loss function
2) A Learning rate for the prediction
3) An additive model to add weak learners to minimize the loss function.

Hyperparameters Tuning for the Gradient Boosting Classifier:

- Learning Rate

This determines the impact of each tree on the final outcome. Gradient boosting classifier works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well.

Lower values would require higher number of trees to model all the relations and will be computationally expensive. The model was trained for a number of the learning rates to find the optimum learning rate. The rates that were tested were from 0.075 which is considered very small to a very high learning rate of 3 as given on Table 3.

	Learning Rate	Accuracy score (training)	Accuracy score (validation)
0	0.075	0.977348	0.976679
1	0.100	0.983433	0.983104
2	0.250	0.996425	0.996127
3	0.500	0.998891	0.998672
4	0.750	0.999456	0.999388
5	0.850	0.999430	0.999339
6	0.950	0.999557	0.999395
7	1.000	0.999588	0.999500
8	1.250	0.999660	0.999561
9	1.500	0.999690	0.999574
10	1.750	0.687197	0.688453
11	2.000	0.343345	0.342658
12	3.000	0.518731	0.517245

Table 3: Learning rates

The optimum learning rates were found to be between 0.85 to 1.25 all having the accuracy score of 0.99 which is seen at both the training phase as well as the validation phase of the algorithm. The best learning rate was thus selected for the implementation purposes.

- n_estimator

The number of sequential trees to be modelled. The gradient boosting classifier is fairly robust at the higher number of trees but it can still overfit at a point. Therefore, the number of this parameter that was set for this purpose was 20 for the simulation purposes.

3. XGBoost

XGBoost is the 'acronyms' for Extreme Gradient Boosting and is an efficient implementation of the stochastic gradient boosting machine learning algorithm. The stochastic gradient boosting algorithm, also called gradient boosting machines or tree boosting, can be used to boost the decision trees in order to calculate accurately predictions for multi-class cases. The reason is that it uses this implementation which proves to be a useful difference than the gradient boosting algorithm that by using this algorithm the overfitting can be dramatically reduced.

For each supervised Model (algorithms), confusion matrices have been implemented, that allows visualization of their performance. In the following sections the main evaluation scores are presented.

The 3 main evaluation scores are:

- **Precision**

Precision = True Positives / (True Positives + False Positives)
Appropriate when minimizing false positives is the focus

- **Recall**

Recall = True Positives / (True Positives + False Negatives)
Appropriate when minimizing false negatives is the focus. Sometimes, we want excellent predictions of the positive class. We want high precision and high recall. This can be challenging, as often increases in recall often come at the expense of decreases in precision

- **F1-Score**

Classification accuracy is widely used because it is one single measure used to summarize model performance. F-Measure provides a way to combine both precision and recall into a single measure that captures both properties. Alone, neither precision nor recall tells the whole story. We can have excellent precision with terrible recall, or alternately, terrible precision with excellent recall. F1-score provides a way to express both concerns with a single score.
Once precision and recall have been calculated the two scores can be combined into the calculation of the F1-Score. The traditional F1-Score is calculated as follows:

F1-Score = (2 * Precision * Recall) / (Precision + Recall)

Like precision and recall, a poor F1- score is 0.0 and a best or perfect F1- score is 1.0

- **Confusion Matrix**

A confusion matrix is a table to understand how well our model predictions have been performed (especially confusing when we have multiple classes and not the classic binary 0/1 problems).

EVALUATION RESULTS

BCOM provided the dataset and the testbed for the aLTEr attacks. The sample dataset contained 809361 observations where 80% of the samples was used for training and 20% of the samples was used for testing purposes. For all the classification reports the incoming traffic has been identified as belonging to the numbers suggesting they belong to the certain class of the packet. The following three packets were used for the classification report purposes.

1. BENING --> 0

2. Dos Hulk --> 1

3. Dos slowloris --> 2

The evaluation of the machine learning models for the testing cases is summarized in the following tables.

3.1.1.3 EVALUATION SCORES FOR RANDOM FOREST MODEL

Packets	Precision	Recall	f1-score	Number of Samples
0	1.00	1.00	1.00	74251
1	1.00	1.00	1.00	61882
2	1.00	1.00	1.00	25740

Table 4: Evaluation scores for Random Forest Model

The confusion Matrix is given as below:

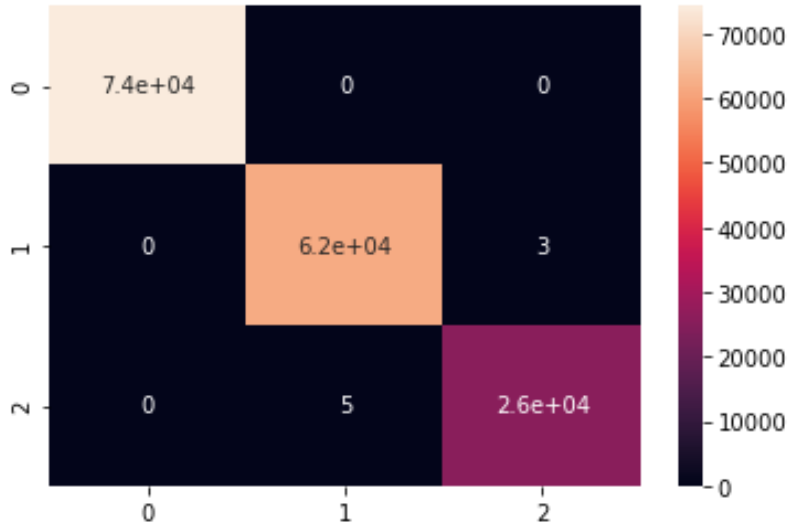


Figure 19: Confusion matrix for Random Forest Model

ROC Curve for Random Forest Classifier

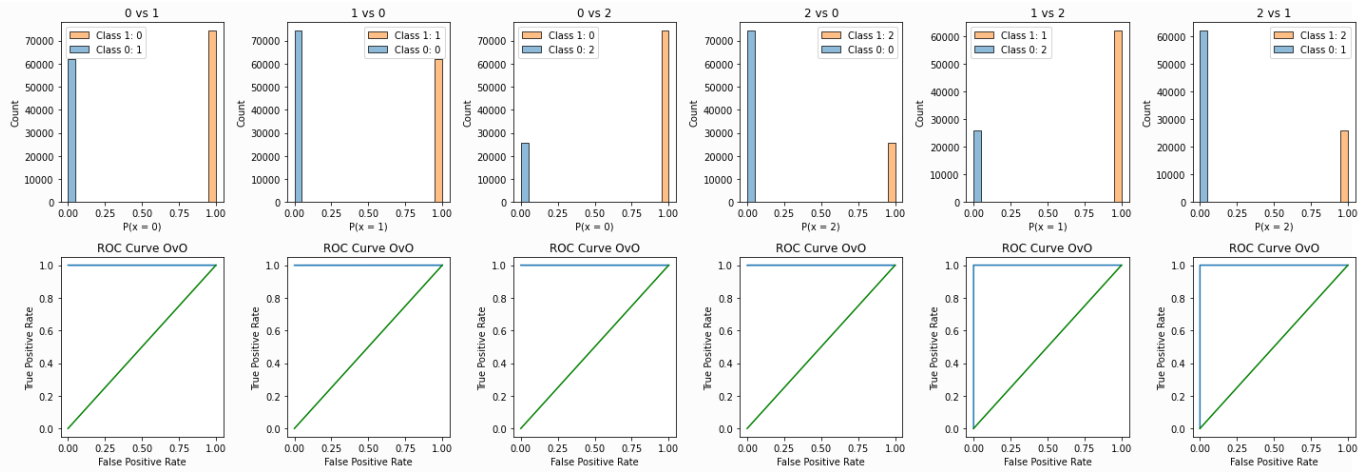


Figure 20: ROC Curve for Random Forest Classifier

The results for the one vs one class for the Random Forest classifier is shown in the Figure 20: ROC Curve for Random Forest ClassifierFigure 20. The average area is 1.00. The same results were obtained when the confusion matrix was evaluated for the said classifier.

0 vs 1 ROC AUC OvO: 1.0000
1 vs 0 ROC AUC OvO: 1.0000
0 vs 2 ROC AUC OvO: 1.0000
2 vs 0 ROC AUC OvO: 1.0000
1 vs 2 ROC AUC OvO: 1.0000
2 vs 1 ROC AUC OvO: 1.0000
Average ROC AUC OvO: 1.0000

3.1.1.4 EVALUATION SCORES FOR GRADIENT BOOSTING CLASSIFIER

Packets	Precision	Recall	f1-score	Number of samples
0	**1.00**	**1.00**	**1.00**	**74251**
1	**1.00**	**1.00**	**1.00**	**61882**
2	**1.00**	**1.00**	**1.00**	**25740**

Table 5: Evaluation scores for Gradient Boosting Classifier

The confusion Matrix is given as below:

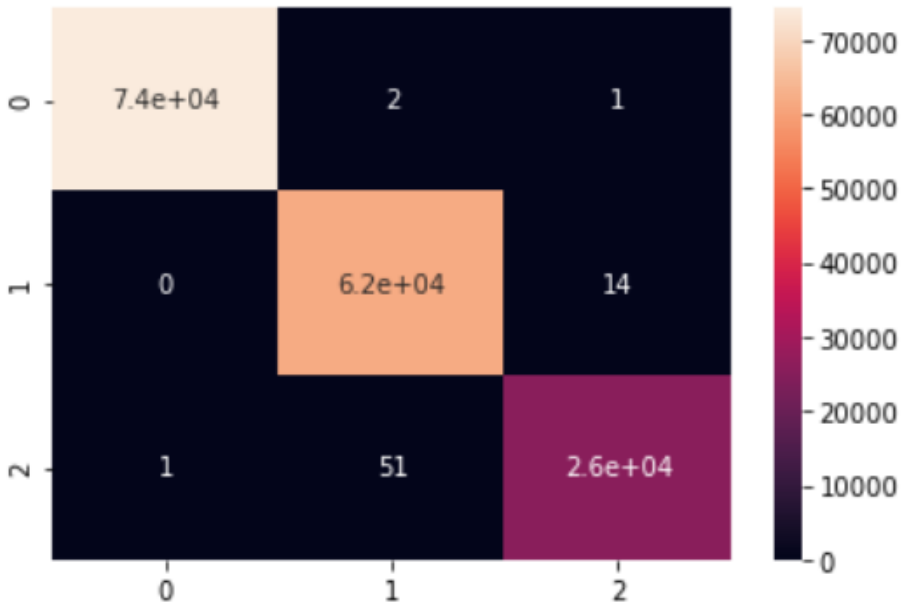


Figure 21: Confusion matrix for Gradient Boosting Classifier

ROC Curve for Gradient Boost Classifier

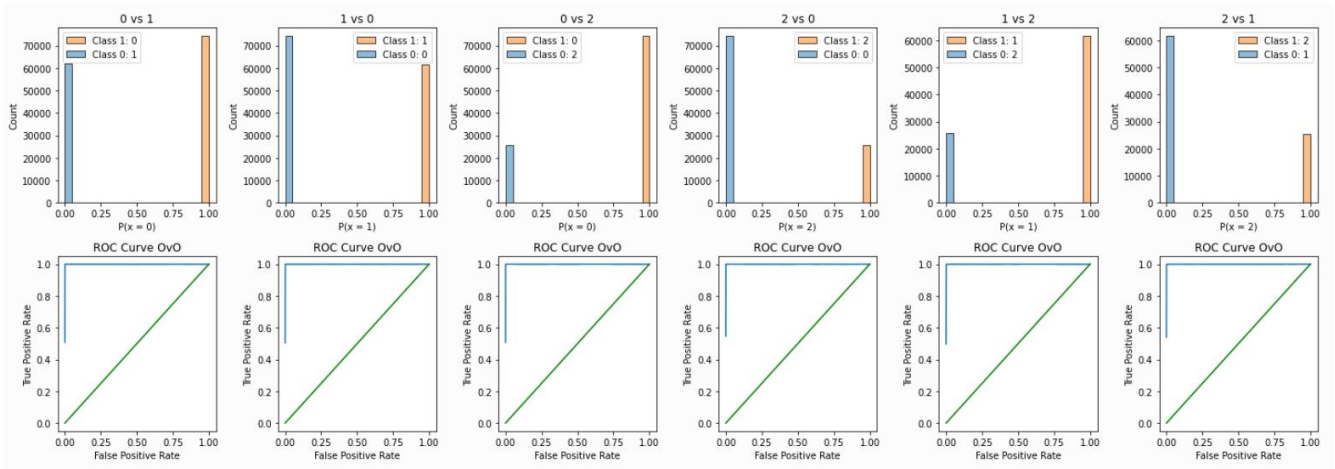


Figure 22: ROC Curve for Gradient Boost Classifier

0 vs 1 ROC AUC OvO: 1.0000
1 vs 0 ROC AUC OvO: 1.0000
0 vs 2 ROC AUC OvO: 1.0000
2 vs 0 ROC AUC OvO: 1.0000

1 vs 2 ROC AUC OvO: 1.0000
2 vs 1 ROC AUC OvO: 1.0000
average ROC AUC OvO: 1.0000

The results for the one vs one class for the Gradient boosting classifier is shown in the above-mentioned figure. The average area is 1.00. The same results were obtained when the confusion matrix was evaluated for the said classifier.

3.1.1.5 EVALUATION SCORES FOR XGBOOST MODEL

Packets	Precision	Recall	f1-score	Number of Samples
0	**1.00**	**1.00**	**1.00**	**74251**
1	**1.00**	**1.00**	**1.00**	**61882**
2	**1.00**	**1.00**	**1.00**	**25740**

Table 6: Evaluation scores for XGBoost Model

The confusion Matrix is given as below:

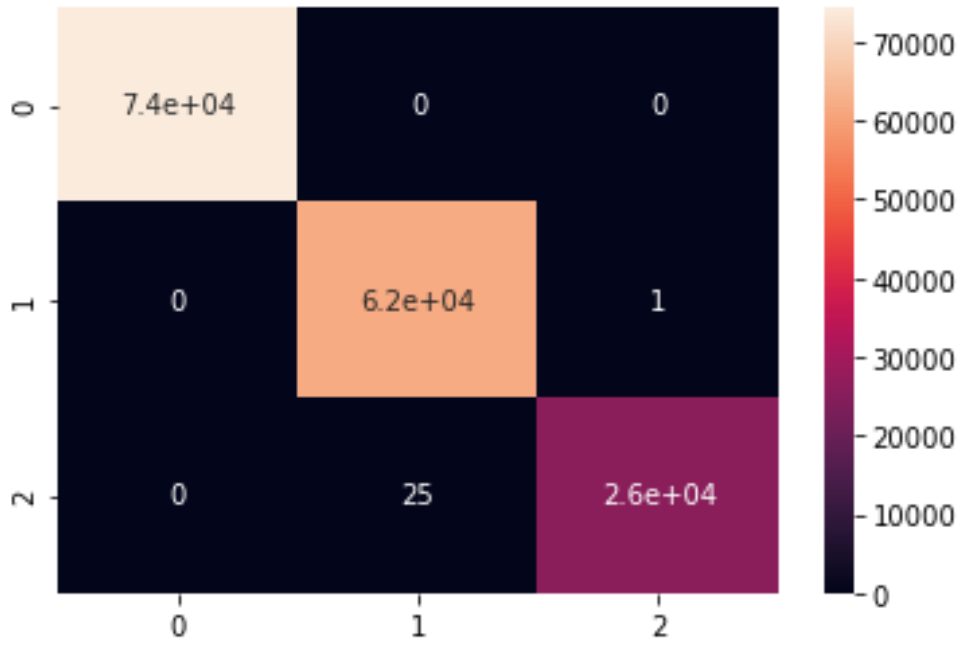


Figure 23: Confusion matrix for XGBoost Model

ROC Curve for XG Boost Classifiers

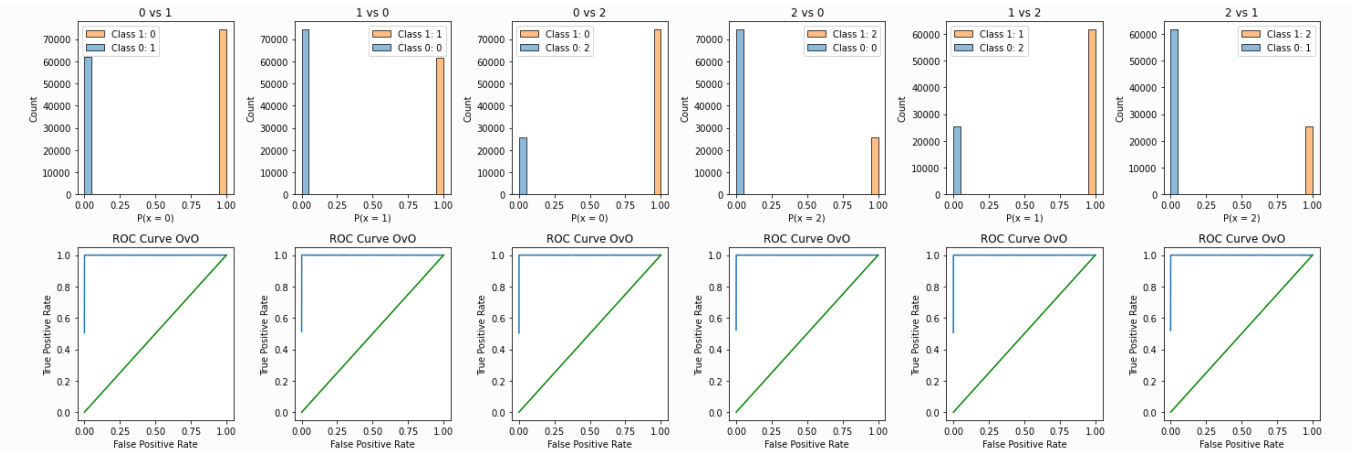


Figure 24: ROC Curve for XG Boost Classifiers

The ROC curves for the one vs one has been shown in Figure 24. The one vs one classes curve for each class show that the classifier is performing according to the expectations with each class is distinguished from the other by the xg boost classifier.

0 vs 1 ROC AUC OvO: 1.0000
1 vs 0 ROC AUC OvO: 1.0000
0 vs 2 ROC AUC OvO: 1.0000
2 vs 0 ROC AUC OvO: 1.0000
1 vs 2 ROC AUC OvO: 1.0000
2 vs 1 ROC AUC OvO: 1.0000
average ROC AUC OvO: 1.0000

The results for the one vs one class for the XG boost is shown for each class. The average area is 1.00. The same results were obtained when the confusion matrix was evaluated for the said classifier.

Finally, in Table 7, the comparison between the Three Classification Algorithms can be seen:

	Trainning Accuracy	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	Testing Accuracy
Gradient Boosting	0.999666	0.000161	0.000439	0.000049	0.999619
XGBoost	0.999896	0.000161	0.000463	0.000049	0.999888
Random Forest	0.999990	0.012674	0.021525	0.007030	0.999945

Table 7: Comparison between the three classification algorithms

The results show that all the models performed well during both the training and testing phase. In particular, Random Forests has shown higher training and testing accuracy, however, the differences are very small and therefore any model can be used for the deployment purposes.

3.1.1.6 MTTD AND MTTR

Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) are commonly accepted as the two main KPIs in incident management. The first indicator measures the average delay between the start of the incident and the time the organization or a system detects or identifies the problem. MTTD can calculated by adding up all the delays between failure and detection, and dividing the total amount of times by the number of occurrence. The failure time starts when the attacker launches the attack, and in the context of the aLTEr scenario, and it is the moment the attacker (MitM) changes the DNS look-up destination IP address, so the issue starts when the malicious DNS messages is transmitted in the UE's GTP tunnel. The detection time includes the following latencies:

- Time To Translate (TTT): from the collection of the raw DNS IP to the translation into a DNS transaction log by the MonB5G MS,
- Time To Analyze (TTA): from receiving logs from the MonB5G MS to the detection of the anomaly by the MonB5G AE
- Time To Detect (TTD) : from the collection of raw data to the incident detection

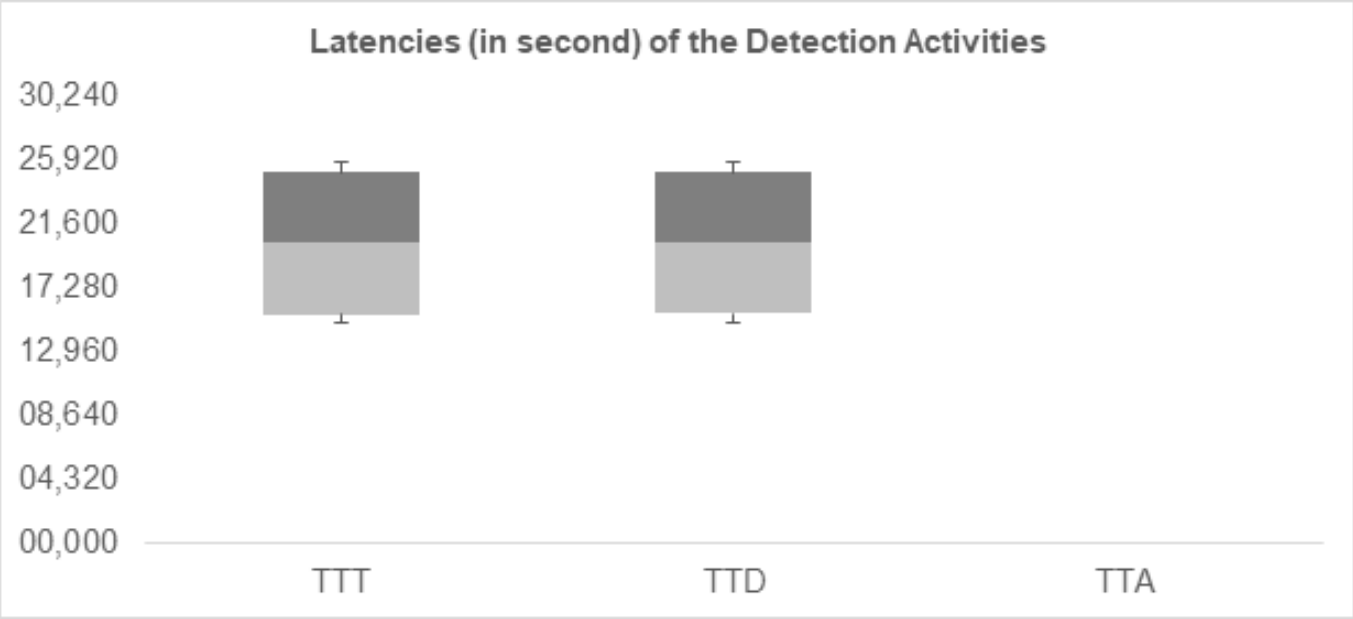


Figure 25: The latencies of the detection activities measured over 30 aLTEr attacks

The monitoring activity, including the collection and transformation of raw data into protocol transactions, appears to have the greatest impact on the detection time.

As with the MTTR measurement, the following transition times are also used in addition:

- the time needed for the MonB5G DE to take the decision to remedy the incident
- the time to deliver the action plan to the ACT
- the execution of the remediation actions

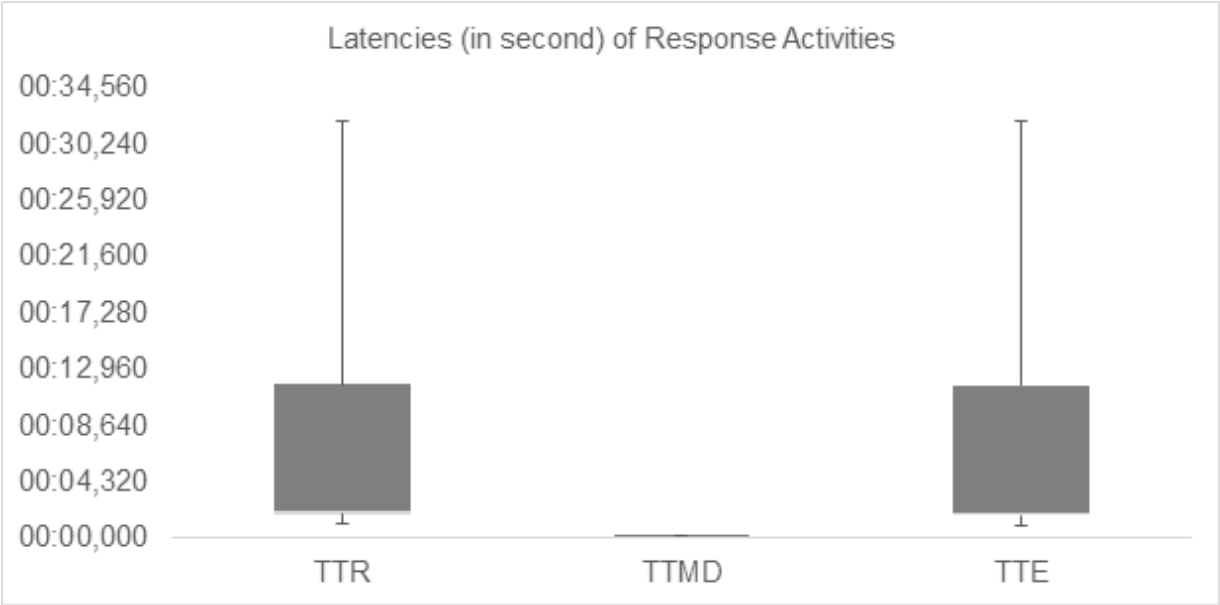


Figure 26: The latencies of the response activities measured over 30 aLTEr attacks

These measures show that the activity-related delays in performing remediation actions are proportionally much greater than those in making decisions.

Finally, we calculated the total duration of incident detection and response activities, from the event of the attack to the execution of remediation actions.

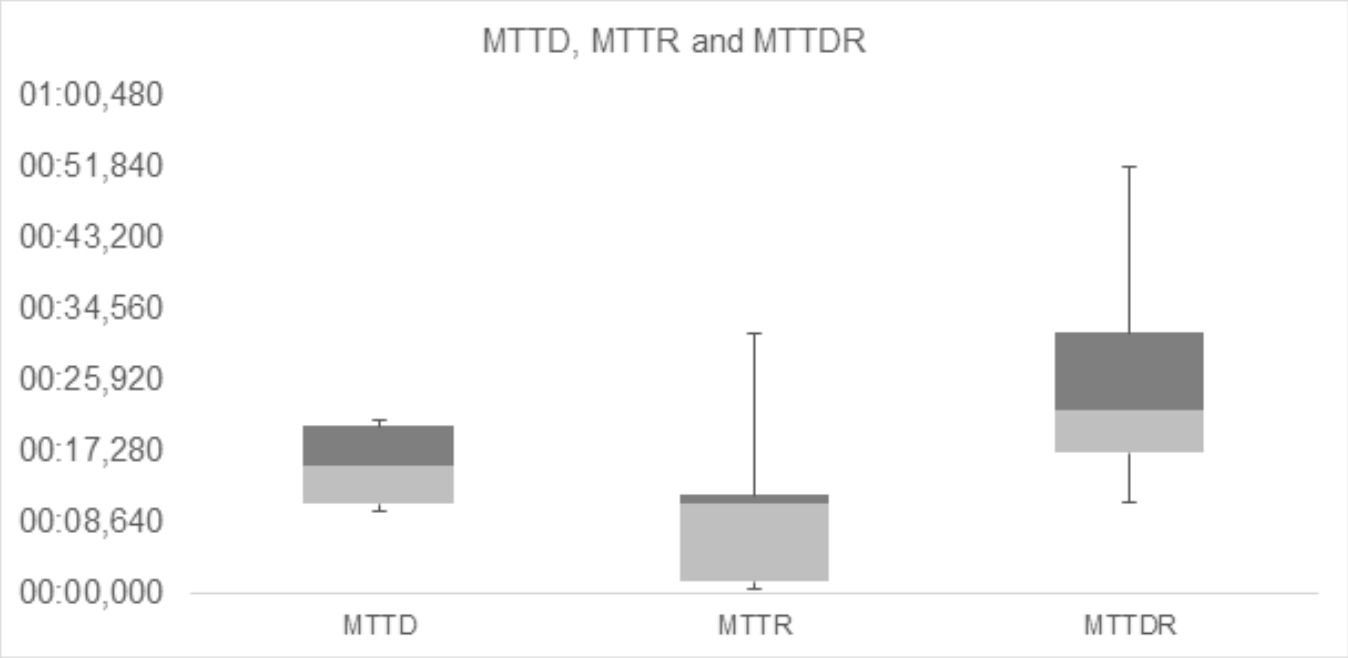


Figure 27: The MTTD, MTTR and MTTDR measured over 300 aLTEr attacks

These measurements show that optimizing the implementation of functions involved in hotspots can improve the reaction time to detect and remediate attacks, these functions are found in the collection and processing of monitoring data, and the execution of action plans. However, these functions are involved at the integration interfaces with the system we aim to defend, and they depends on the attack scenario we plan to address.

These measurements show that optimizing the implementation of functions involved in hotspots can improve the reaction time to detect and remediate attacks, these functions are found in the collection and processing of monitoring data, and the execution of action plans. However, these functions are involved at the integration interfaces with the system we aim to defend, and their performance depends on the attack scenario we plan to address and the API that are used in the remediation actions.

When focussing on the data analysis and decision making activities of the AE and the DE, the measurements show very fast processing time as shown in Figure 28, which achieve the KPIs of the Scenario 1:

- 10x faster identification of security attack/anomaly;
- 10x faster attack remediation and reconfiguration in the order of 10s.

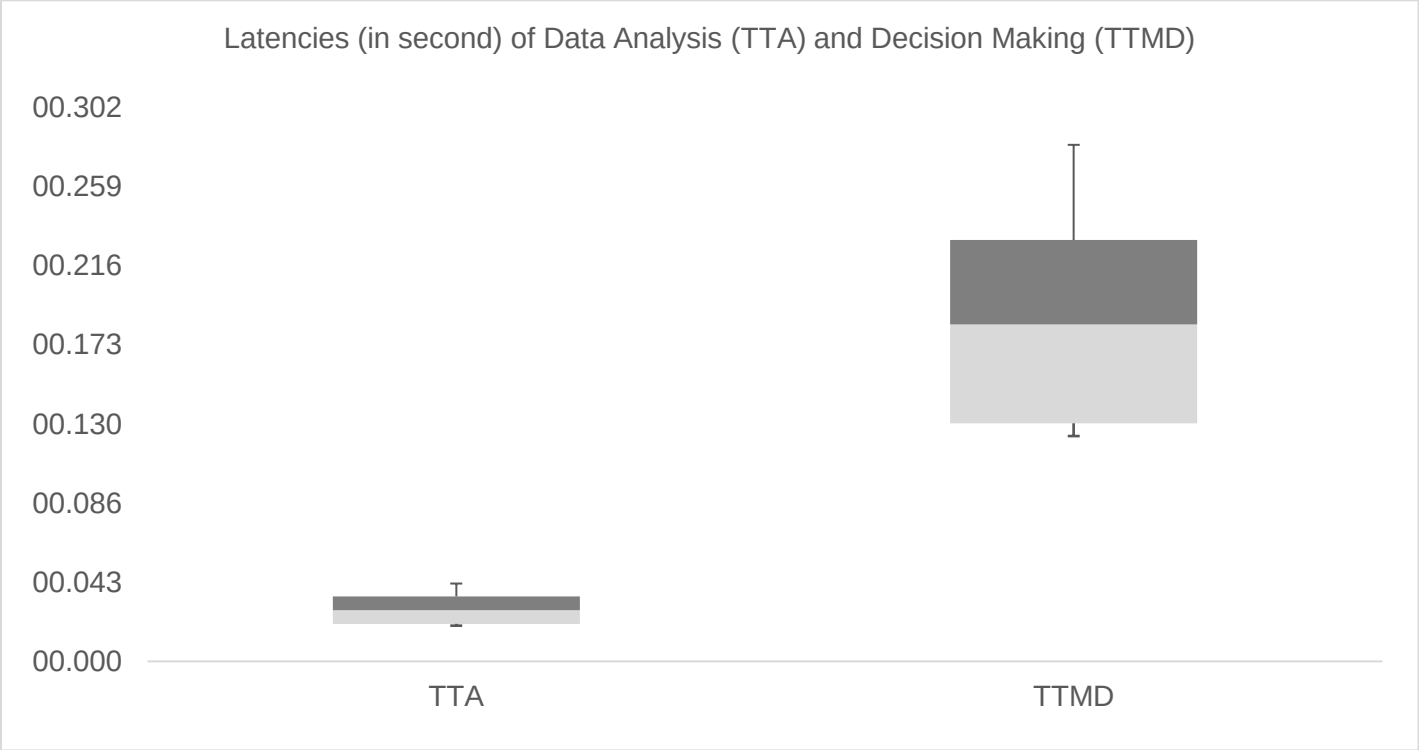


Figure 28: Time To Analyse and Time to Make a Decision for 30 aLTEr attacks

3.1.2 SCENARIO MMTC ATTACK

Our work aims to provide a solution featuring zero-touch security management of in-slice attack detection and mitigation considering mMTC slices in 5G [MMTC]. The proposed solution relies on machine learning to detect abnormal traffic of MTC devices that could cause DDoS on the control plane of the 5G core network (by flooding the AMF with signalling messages. We focus on protecting the AMF as it is the entry point of the 5G CN and treats all the Attach Requests coming from the different gNB under its control. The closed-loop control is composed of three entities: MS, which collects information from the AMF, AE, which uses ML to predict attacks, and DE, which reacts to the alert sent by AE by acting on AMF (block and blacklist UE). In the proposed framework, MS monitors the Attach Request received by the AMF, while DE requests AMF to send Registration Reject to suspected devices.

3.1.2.1 MEASUREMENT METHODOLOGY

To evaluate the performance of the proposed framework, we focus mainly on the performance of the attack detection algorithm, which is the key function of the closed-loop control. To this end, we evaluate the Gradient Boosting algorithm to detect DDoS attacks accurately and compare its performance with a statistical method. Like the Gradient Boosting solution, the statistical method is applied at the end of the event. Based on the event duration, the statistical method defines a limit function using the mathematical function of β(3,4) to compare the different points by the report to this limit; all the points exceeding this limit are considered as a possible attack. The main differences compared to Gradient Boosting are: (i) it needs the exact duration of the event; (ii) it uses the β(3,4) function to deduce the limit.

We measure the accuracy of the Gradient Boosting algorithm in front of normal and abnormal traffic. On normal traffic, the accuracy denotes how often the system yields a detection rate of UEs that is greater than zero. This does not mean that these devices will get banned, but ideally, a value of 0 should be returned for all the devices emitting normal traffic. To evaluate our model on normal traffic (True Positive (FP) = False Negative (FN) = 0), we generate data for 500 normal events and run our detection algorithm on each of them. We then counted the number of UEs for which we obtained a greater-than-zero detection rate versus the total number of UEs in all the events. We generate event durations randomly (between 30 and 250 seconds). Regarding malicious traffic (True Negative (TN) = False Positive (FP) = 0), we also ran 500 tests, but this time, between 7 and 15 Attach Requests are received every 6 seconds, for a total duration that is random between 30 and 250 seconds.

3.1.2.2 KPI DESCRIPTION AND RESULTS

AE: Gradient Boost

Figure 29 allows visualizing the results for normal traffic. The points correspond to the event data, while the green line is the anomaly interval. If a point is outside the limit (in green), it will be assumed as an attack. For normal traffic, the accuracy is computed as (1 − FP)/ (T N+FP). Hence, the results show an Accuracy on normal traffic of 96.76 %. We expected this result as the interval used in our training data includes around 95% of the data in the training dataset. Figure 30 illustrates the results for malicious attacks. For this case, the accuracy is computed as (1− FN)/ (FN+T P). Hence, the results show an accuracy of 96.2 %. This represents an excellent result as banning a relevant part of the devices taking part in a DDoS attack is enough to mitigate it. We recall that this is just the detection rate calculated by the AE component, the final decision regarding the devices that should be banned is taken by the DE component.

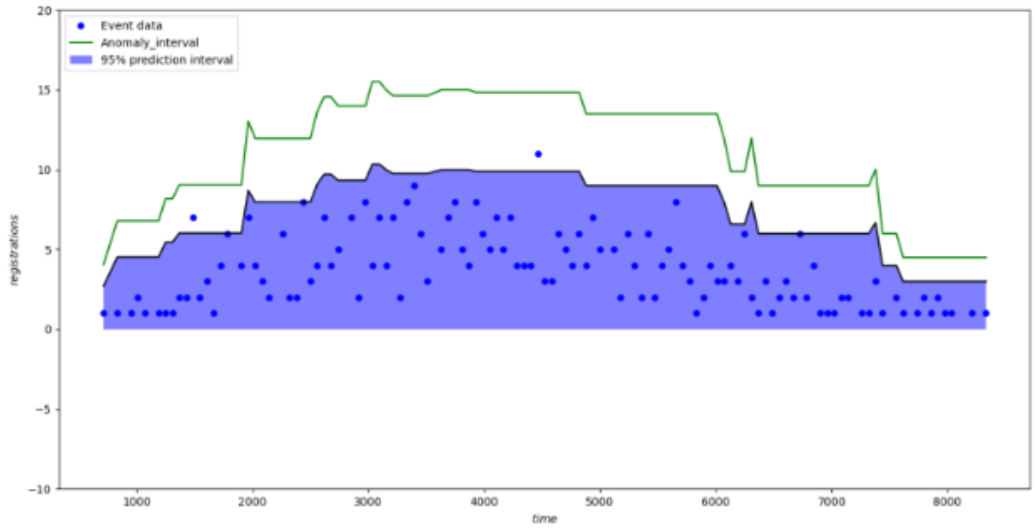


Figure 29: Result of the detection algorithm over normal traffic.

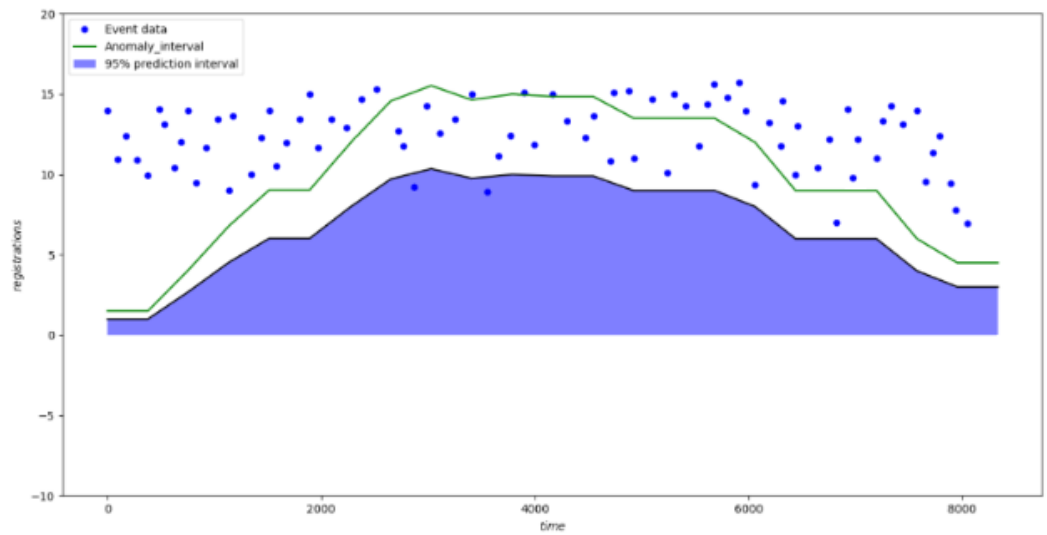


Figure 30: Results of the detection algorithm over abnormal traffic.

Table 8 shows the performance of the Gradient Boosting-based solution when modifying the AE DETECTION T HRESHOLD value. It is worth recalling that this value is used to derive the detection rate and corresponds to a protecting gap to reduce the impact of the ML prediction and hence reduce the FP value. We remark that the value allowing to reduce both FP and FN is 2.0. Also, when the AE DETECTION THRESHOLD value increases, FP is reduced as the FN increases, whereas when it is reduced, both FP and FN increase.

AE_DETECTION_THR.	1.2	2.0	3.0
Normal event	82.98654	96.76859	97.64853
Abnormal event	92.92134	97.64643	75.7584

Table 8: Impact of the AE detection threshold on the Gradient Boosting accuracy.

AE: Statistical Method

For the sake of comparison, we used the same scenarios as for Gradient Boosting to generate normal and abnormal traffic. Then, we applied the statistical method and verified its accuracy in detecting attacks. Figure 31 illustrates the usage of the statistical method in case of an abnormal event. The discontinue green line shows the β(3,4) curve obtained according to the event duration. The β(3,4) curve allows us to have a limited path from which all the outside points are considered anomalies, hence potential attacks. The statistical method's results show that 36.0 % of the Attach requests are not following the β(3,4) distribution (they are out of the limit path). Therefore, they can be considered potential attacks. On the other hand Figure 32 presents a test of a normal event. The results show that 90.0 % of the Attach Requests follow the β(3,4) distribution. The accuracy of the statistical method to detect anomaly are: ((96% (normal traffic), 97% (abnormal traffic))). We remark that these values are weaker than the ones obtained with the Gradient Boosting algorithm. We argue these differences by the fact that the duration estimation has a strong impact on the statistical solution. The shape of the β(3,4) curve changes drastically according to the duration (noted D), which seriously impacts the accuracy. For instance, we change the duration by e = 2 sec, and the obtained results are summarized in Table 9. We see clearly from this table the impact of the duration on the accuracy as a small error on the duration drastically yields a drop in the accuracy. Particularly, if the duration is less than the real one, many points will be out of the curve. In the Gradient Boosting algorithm, we do not have this concern, as the latter normalizes the sample period duration and uses the trained model to detect the interval.

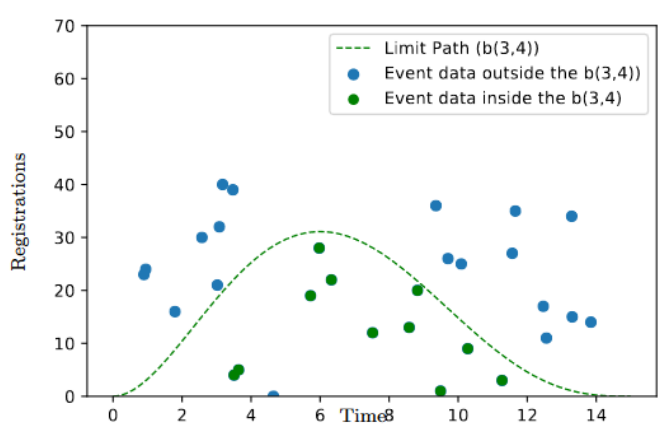


Figure 31: The result of the statics method over abnormal traffic.

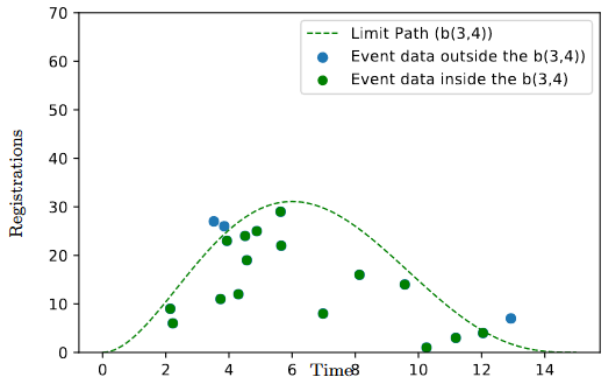


Figure 32: The result of the statics method over normal traffic.

Method	GB	Static		
		D - Δt	D	D + Δt
Normal traffic	96.76859	45.18924	84.21052	80.24568
Abnormal traffic	97.64643	30.4156	57.142857	51.86854

Table 9: The accuracy of the statistical model considering different Duration values.

DE:

Regarding DE performances, we evaluated the first version that relies on a single threshold. Accordingly, in this section, we evaluate the impact on the number of blocked devices. Table 10 shows the number of banned UEs for three values of the thresholds. As expected, we remark that lower threshold values (ex. 0.1) are very conservative, which allows blocking more UE. While a higher threshold value (ex. 0.8) may be less strict and reduce the number of banned UE. We advise two solutions to fix the threshold value. The first one considers the performance limit of the element to protect against DDoS attacks. In our case, we computed the response time of the AMF to Attach Requests while increasing their number. After a certain number of Attach Requests, we noticed that the AMF started to be very slow, which can be caused by a DDoS attack. Therefore, after some tests, we found that the value of threshold equal to 0.35, which avoids reaching the number of Attach Request that yields bad AMF performances. Another solution is to use a dynamic threshold value that decreases or increases over time according to the number of consecutive events classified as an attack.

DE_DETECTION_THR.	0.1	0.35	0.8
Nb (Normal event)	3	1	0
Nb (Abnormal event)	21	16	10

Table 10: Impact of the DE Detection threshold

3.1.2.2.1 KPI 2: 10X FASTER ATTACK REMEDIATION AND RECONFIGURATION IN THE ORDER OF 10S

Since of the event duration in IoT are less than 10s, we are sure with our solution that the detect is done less than 10s. Moreover, our AE + DE take 5s in order to detect the DDoS attack (Detection and mitigation).

3.1.2.2.2 END TO END SLICE AVAILABILITY

Since the precision is about 97%, it means that only3% are missed, so, we have the ability to sustain 97% of SLA (availability). But knowing that not all the 3% will lead the system distribution. We can say that we are able to sustain 97% of the service.

3.1.3 SCENARIO FEDERATED LEARNING ATTACK

In this section, we present a novel framework to automatically detect malicious participants in FL process. First of all, based on real test bed, we generate a realistic dataset that focuses on the latency as service-oriented KPI of running network slices. This dataset is then exploited to build an analytic function about the latency prediction, in federated way. In addition, the basic idea of our framework is to select dynamically one participant as trust entity, by leveraging deep reinforcement learning. The trusted participant will be then in charge of identifying poisoning model updates, using unsupervised machine learning. The main contributions of this work are summarized as follows:

- We use OAI[12] to generate a real dataset about the latency KPI of the AMF, as a VNF. This service-oriented KPI corresponds to the response time to handle UEs attach requests, when considering different configurations, such as available RAM memory and number of CPU (see Section 4 in the MonB5G deliverable D5.3 [D5.3]).
- Exploiting our dataset, we build a DL-based model in federated way between several running network slices. Our model enables to predict the latency of the AMF function and hence anticipating any latency-related SLA violation.
- We also build an online DRL model that selects dynamically a network slice as a trust participant, at each federated learning round, i.e., when participants send their local models towards the central node, based on several metrics such as their reputations.
- At each FL round, the trusted participant applies dimensionality reduction scheme and unsupervised machine/deep learning to detect the poisoned model(s) and hence malicious participant(s).

3.1.3.1 MEASUREMENT METHODOLOGY

We developed the main components of our framework, in terms of FL-based latency model, DQN-based participants' selection model, and clustering and dimensionality reduction-based attack detection model, using TensorFlow Python library and leveraging our EARCD dataset. We evaluate our FL-based model to predict the latency KPI, in terms of Mean Squared Error (MSE) on top of two different weight optimizers (SGD and ADAM), in order to determine the suitable optimizer, that improves the prediction accuracy of our both FL-based and attack detection models.

We note that we evaluate our FL-based model in and without presence of malicious nodes, in order to show the impact of poisoning attacks on the performance of our FL model. The malicious nodes (network slices) generate poisoning attacks by introducing new malicious/incorrect data samples and build their local models on top of such data. For example, the malicious nodes may consider high latency values, even when receiving a low number of attach requests and vice versa.

[12] https://openairinterface.org/

In addition, we trained our DQN model during more than 5000 learning episodes. Once converged, we deployed our DQN-based model at the inter-domain slice manager, which is in charge to select a trust participant at each FL round. Moreover, to evaluate our attack detection scheme, after a given number of FL training rounds (r < R', the ten network slices' DSMs send their model updates to the inter-domain network slice manager. The latter first selects one network slice as a trust party, before sharing with it the model updates. Table 11 gives more details about parameter settings used in our simulation.

The parameter settings	**Value**
Federated Learning (FL)	
Number of layers	4
Number of neurons	20
Optimizer 1	SGD
Learning rate	0.0001
Optimizer 2	ADAM
Activation function	ReLU
Loss function	MSE
Reinforcement Learning (DQN)	
Number of layers	2
Number of neurons	24
Optimizer	ADAM
Learning rate	1e-2
episodes	5000
Dimensional Reduction Algorithm (DRA)	
Dimension of LDA	2D
Dimension of PCA	2D
Clustering Algorithm (CA)	
Number of cluster (k)	1, 2

Table 11: The parameter settings

3.1.3.2 KPI DESCRIPTION AND RESULTS

3.1.3.2.1 Evaluation of Latency prediction in Federated way:

Following, we present the results without malicious nodes, for two different optimizers: "ADAM" and "SGD". Figure 33 depicts the MSE metric of our FL model during several FL rounds, with and without presence of malicious network slices. On top of the ADAM optimizer, we clearly observe that MSE of our model without malicious nodes is lower than with malicious nodes, around 0.1035681. However, it increases as we add more malicious nodes that will inject more malicious/incorrect data in terms of latency KPI. Therefore, poisoning data attack affects highly not only the performance of our FL model in terms of MSE, but also the model convergence towards an accurate global model.

Similarly, Figure 34 shows the MSE metric of our FL model on top of the SGD optimizer. We also see that the MSE increases as the number of malicious network slices is increased as well, and the FL model without malicious nodes outperforms the other models (with malicious nodes), with a MSE of 0.173584. Hence, these results confirm the results of Figure 33 and show that data poisoning attack can have a negative impact on

the performance of our FL model, especially in terms of global model convergence, whatever the used learning optimizer (ADAM or SGD).

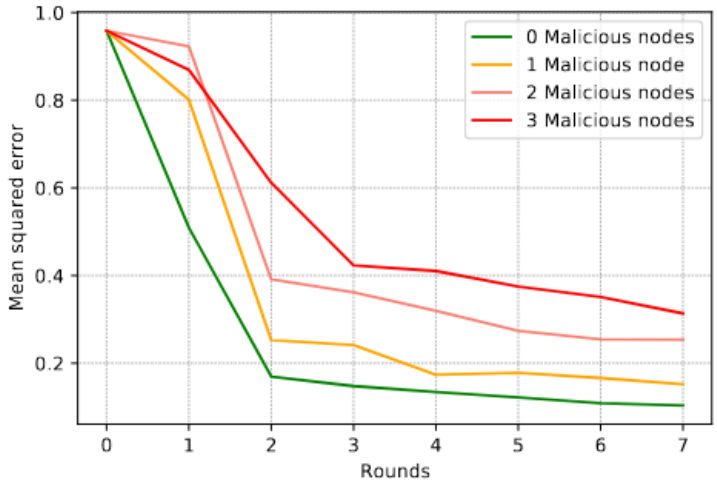


Figure 33: Mean Squared error of our FL model when using ADAM Optimizer

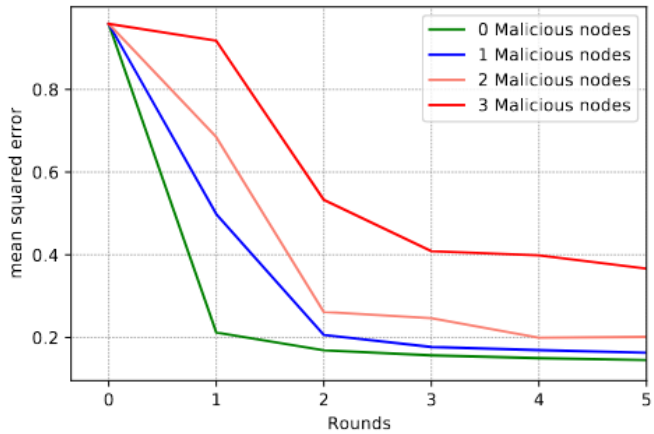


Figure 34: Mean Squared error of our FL model when using SGD Optimizer.

3.1.3.2.2 Evaluation of Trust Participant Selection:

Table 12 shows the results of the selected DSM based on our DQN-based scheme, for two different time instances t1, and t2. These results are obtained based on the nodes' reputations (Repi €[0,1]) . As we observe, when t = t1, the DSM 4 was selected as trust node, since it maximizes the reputation value. Similarly, when t = t2, the network slices' DSMs have different reputation values, however, DSM8 is selected, because it presents the highest value. In other words, at each FL round, our DQN-based algorithm enables to select the network slice maximizing the reward, since the reputation value is determined based on the reward.

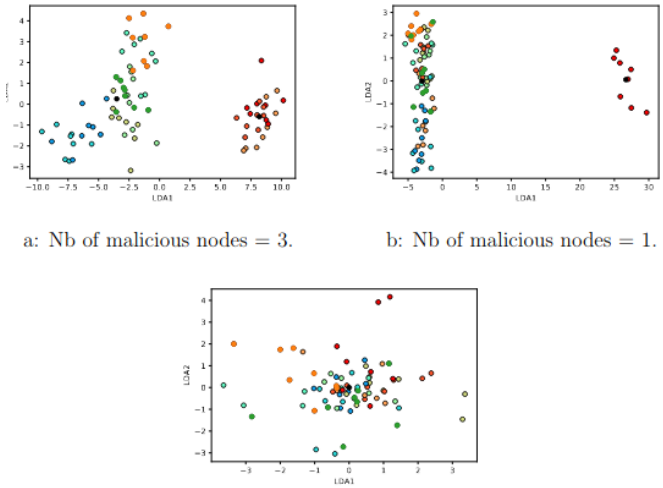
$t : t1$									
RepDSM1	RepDSM2	RepDSM3	RepDSM4	RepDSM5	RepDSM6	RepDSM7	RepDSM8	RepDSM9	RepDSM10
0.2	0.4	0.3	0.7	0.61	0.48	0.37	0.68	0.54	0.62
Selected DSM			✓						
$t : t2$									
RepDSM1	RepDSM2	RepDSM3	RepDSM4	RepDSM5	RepDSM6	RepDSM7	RepDSM8	RepDSM9	RepDSM10
0.1	0.22	0.15	0.74	0.58	0.52	0.31	0.8	0.60	0.71
Selected DSM							✓		

Table 12: The results of the selected DSM based on our DQN-based scheme

3.1.3.2.3 Evaluation of Data Poisoning Attack Detection

After showing the negative effect of the data poisoning attacks on the performance of the global FL model, in this subsection, we evaluate the performance of our combined dimensionality reduction and unsupervised clustering scheme against data poisoning attacks, and on top of the two different optimizers: ADAM and SGD. Noting that for data poisoning attacks, the malicious node tries to inject incorrect latency values, e.g. high latency value, even when the node has high resource capacity, in terms of memory and computing.

Combining LDA with k-means: Figure 35 and Figure 36 depict the detection results when combining LDA and K-means, on top of ADAM and SGD optimizers, respectively. We also vary the percentage of malicious network slices' DSM and show the trusted nodes that are selected at each FL round (nodes in green colour). As we see, our scheme can clearly detect the malicious nodes, even with only 10% of malicious nodes, i.e., only one malicious node (NB of malicious node = 1). Specifically, the trusted node applies both LDA and K-means, and then all nodes that are in the same cluster with it, are considered as correct models, while the nodes (models) that are in the other (s) cluster (s) will be considered as malicious. Therefore, our trust participant selection algorithm helps us not only to select a trust node, but also to determine malicious nodes when performing the dimensionality reduction and unsupervised clustering. Moreover, determining the trusted cluster of nodes will also help the FL server (IDSM) to select a trust participant for the next FL round.

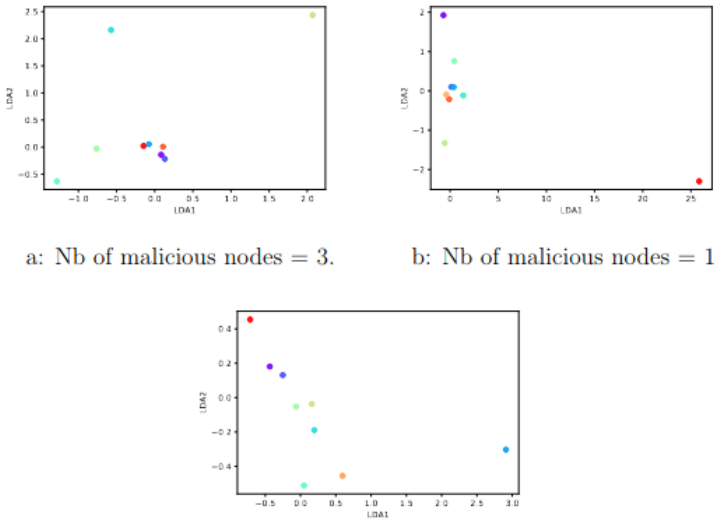


a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 35: LDA + K-means for different number of malicious nodes (ADAM optimizer).



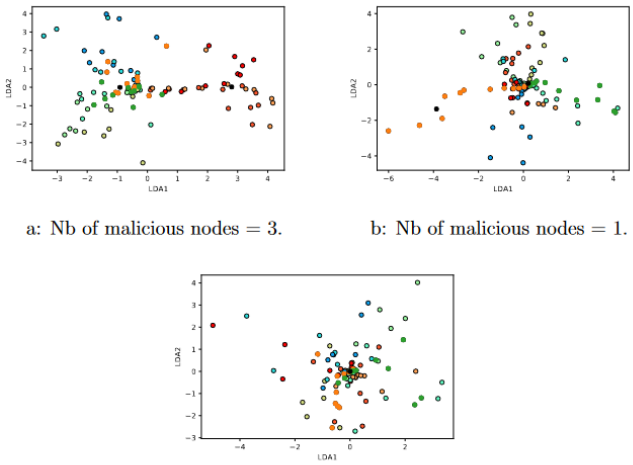
a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 36: LDA + KNN for different number of malicious nodes (ADAM optimizer).

Combining LDA with KNN: Figure 37 and Figure 38 also show the clustering of local models when applying both LDA and KNN, on top of ADAM and SGD optimizers, respectively. Whatever the number of malicious nodes, we also observe that there are always some isolated points, which represent the infected models sent by the malicious DSMs. However, for the LDA technique, we see that the isolated models (infected) are identified better with the ADAM optimizer than with the SGD optimizer (Figure 37 and Figure 36). Hence, the LDA (with KNN or k-means) technique gives better detection on top of the ADAM optimizer.

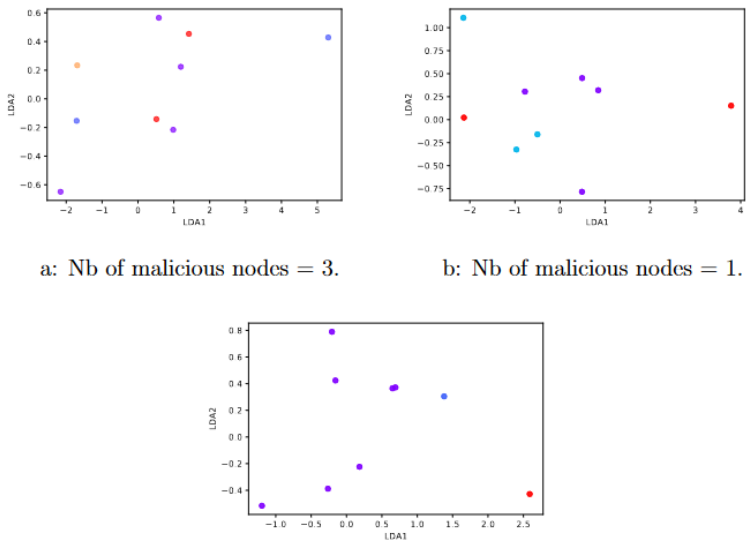


a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 37: LDA + K-means for different number of malicious nodes (SGD optimizer)



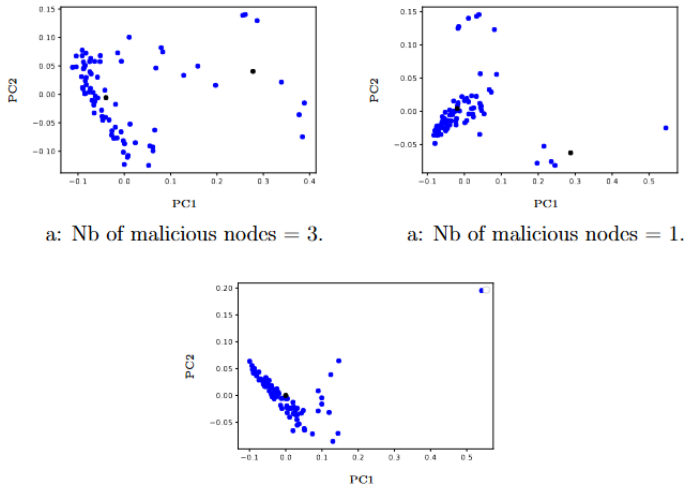
a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 38: LDA + KNN for different number of malicious nodes (SGD optimizer

Combining PCA with K-means: As we did for LDA, we also evaluate the performance of PCA technique when combined with clustering unsupervised algorithms. Figure 39 and Figure 40 show the clustering detection when combining PCA with K-means, on top of the ADAM and SGD optimizers, respectively. We remark that both optimizers succeed in separating and identifying infected models by incorrect data. However, infected models are better identified on top of the SGD optimizer, as compared to the ADAM optimizer. Thus, PCA with K-means gives better performance in detecting infected models on top of the SGD optimizer.

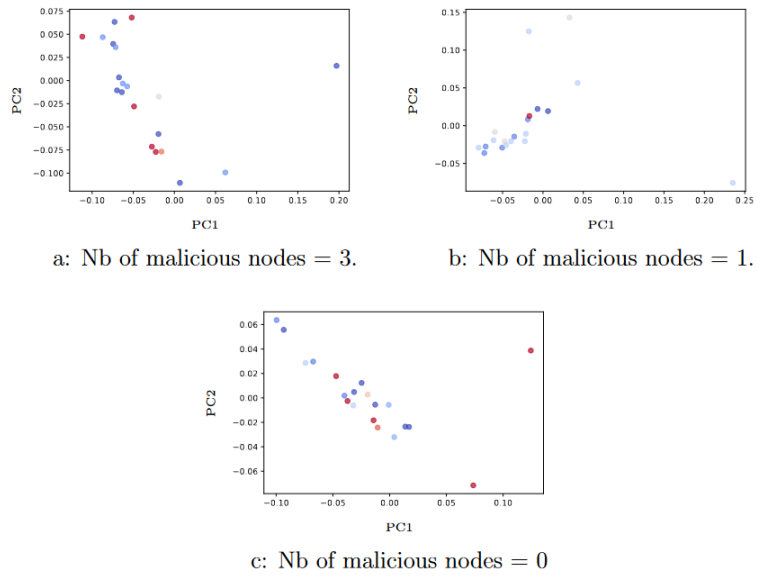


a: Nb of malicious nodes = 3.

a: Nb of malicious nodes = 1.

a: Nb of malicious nodes = 0

Figure 39: PCA + k-means for different number of malicious nodes (ADAM optimizer)



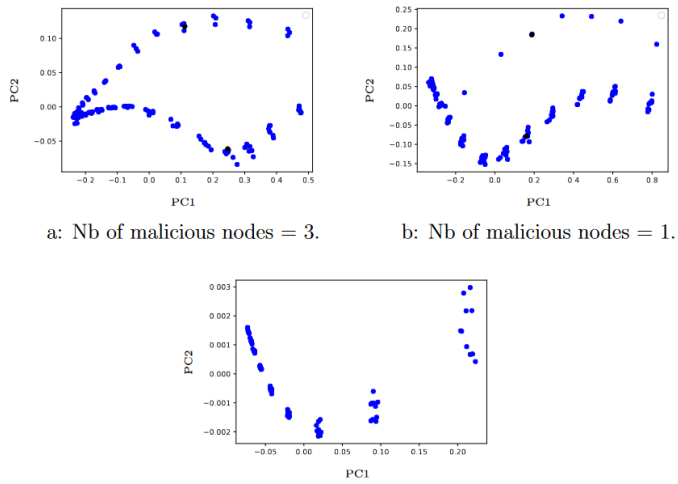
a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 40: PCA + KNN for different number of malicious nodes (ADAM optimizer).

Combining PCA with KNN: Similarly, Figure 41 and Figure 42 depict the detection when combining PCA with KNN on top of the ADAM and SGD optimizers, respectively. As in Figure 39 and Figure 41, PCA with KNN on top of both optimizers clearly separate correct local models from infected ones and thus enable to detect/identify malicious DSMs. We also see that infected models are better identified when leveraging the SGD optimizer than the ADAM optimizer. Therefore, the PCA technique with either K-means or KNN gives better detection of malicious models on top of the SGD optimizer, which is confirmed in, Figure 41.



a: Nb of malicious nodes = 3.

b: Nb of malicious nodes = 1.

c: Nb of malicious nodes = 0

Figure 41: PCA + k-means for different number of malicious nodes (SGD optimizer).

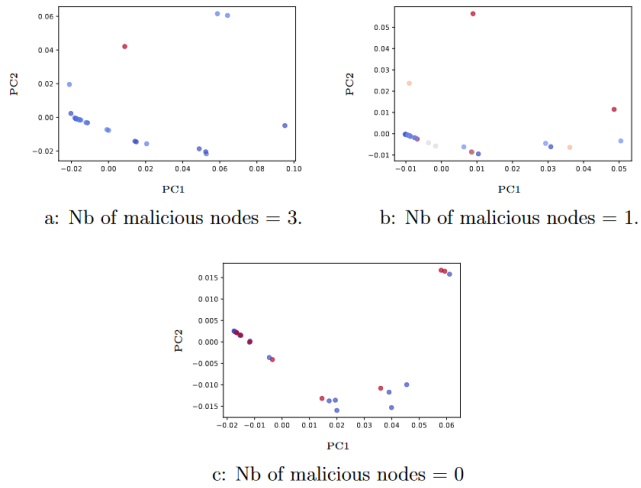


Figure 42: PCA + KNN for different number of malicious nodes (SGD optimizer).

3.1.3.2.4 Evaluation of Model Poisoning Attack Detection

In this subsection, we evaluate our scheme against model poisoning attacks, where malicious DSMs try to send incorrect learning models, which are generated randomly for instance. Figure 43 and Figure 44 show the results when combining LDA and K-means on top of the ADAM optimizer. We remark that LDA with K-means does not clearly succeed to identify the malicious models, Figure 43. However, when combining PCA and K-means on top of the SGD optimizer, the malicious models are better detected, as depicted in Figure 45. Therefore, we can deduce that model poisoning attacks are better detected and identified, when using PCA with K-means techniques, as compared to LDA and K-means combination.

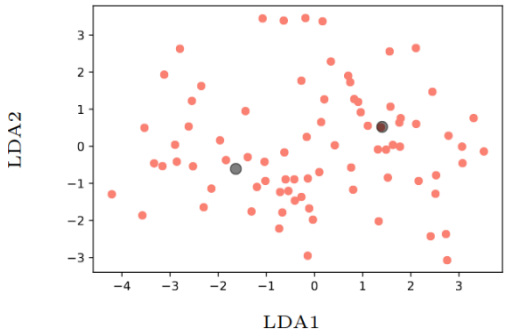


Figure 43: LDA + K-means on top of the ADAM optimizer with the malicious nodes

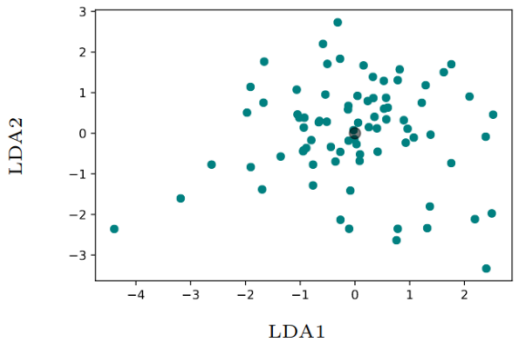


Figure 44: LDA + K-means on top of the ADAM optimizer without the malicious nodes

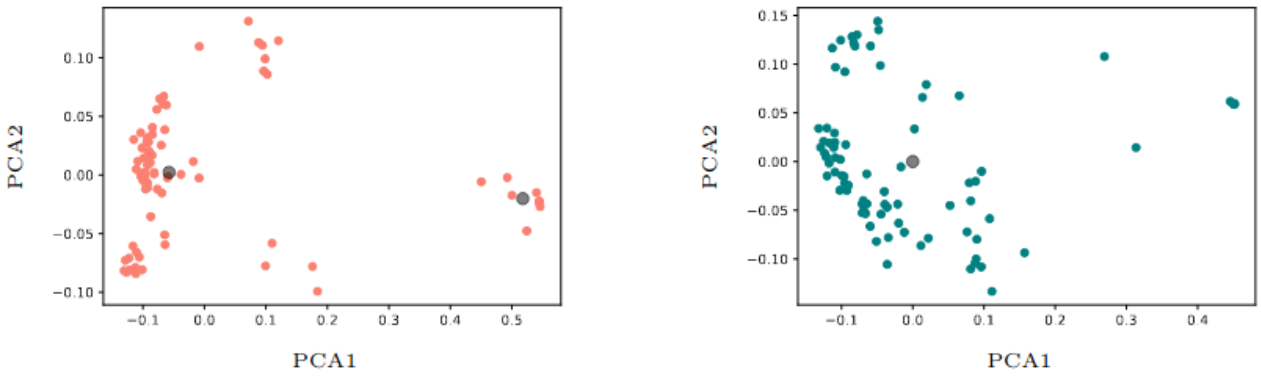


Figure 45: PCA + K-means on top of the SGD optimizer with the malicious nodes and without the malicious nodes

3.1.3.2.5 Impact of malicious nodes detection on the FL accuracy

In Figure 33 and Figure 34 we showed that poisoning data attack affects highly not only the performance of our FL model in terms of MSE, but also the model convergence towards an accurate global model. Figure 46 depicts the MSE metric of our global FL model, when applying our combined dimensionality reduction and clustering scheme, to detect and remove infected models. Noting that we apply our scheme after a given number of FL rounds r = R' = 4, i.e., after collecting some updates from the network slices' DSMs. We clearly observe that our scheme improves MSE of the global FL model, which decreases as we increase the number of FL rounds, even when injecting some infected models from the fourth round. Therefore, our detection scheme enables not only to identify malicious DSMs, but also improving the accuracy of the global FL model.

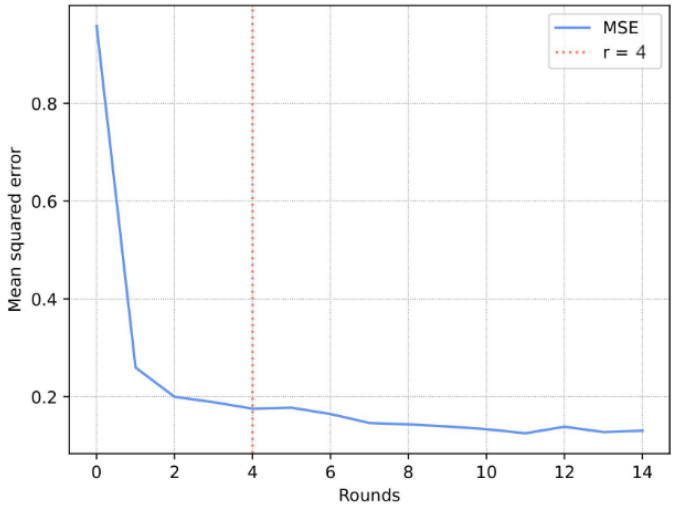


Figure 46: Mean Squared error of the FL global model (ADAM optimizer).

Our solution detects and mitigate the poisoning attacks that is done less than 10s. Actually, our AE + DE take 6s in order to prevent the attacks.

In general, we can deduce that our scheme succeeds to provide stable performance in dealing with both data and model poisoning attacks that can target federated learning-based models. In particular, combining dimensionality reduction and clustering unsupervised learning helps us to not only detect/identify infected learning models, but also to improve the global accuracy of the FL model.

3.2 Energy-based Orchestration

3.2.1 ENERGY-AWARE, MULTI-DOMAIN ORCHESTRATION

The objective of PoC 1 is to optimise the placement of a network slice VNFs locally (intra-domain) for lower energy consumption and latency. We evaluated our proposed solution through model validation and simulation while demonstrating its ability to jointly reduce average service latency by 103.4% and energy consumption by 17.1% compared to a centralized RL solution. Unlike similar works in the literature, the examined PoC 1 solution is decentralized, eliminating any central point of failure and enabling scalability.

The physical network infrastructure was represented as a graph with nodes and edges, representing the sets of the nodes and links respectively (Figure 47). To accommodate the functions of an SDN-NFV enabled network, the nodes are split into two types. One is dedicated to network switching that is responsible for forwarding the service traffic and the other one has the ability, not only to forward but also to instantiate, terminate or migrate VNFs, up to its physical capacity. Both types of nodes handle the routing of the SFC traffic.

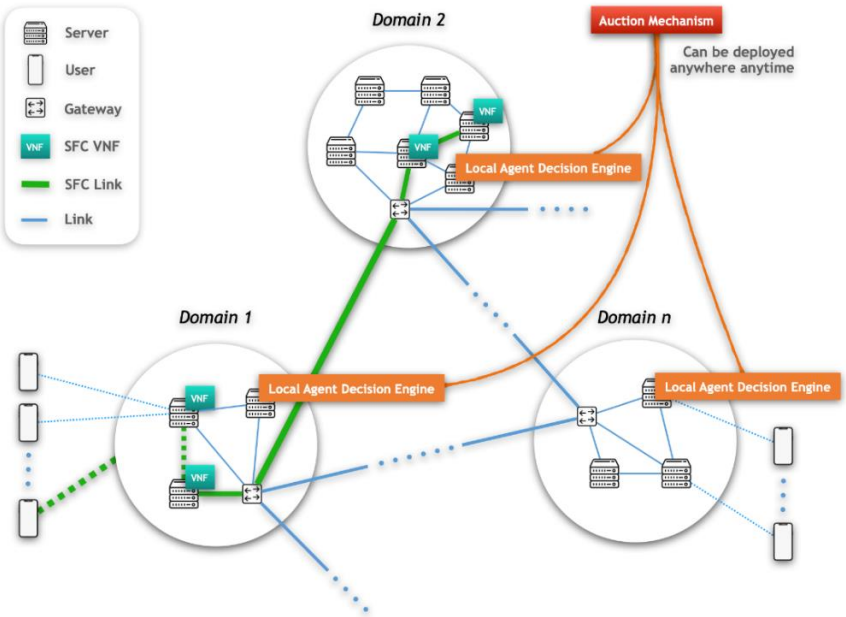


Figure 47: Network graph representation

The network has a finite number of resources that can be measured and tracked through metrics, such as: latency, bandwidth, CPU cores, RAM memory and disk storage. The instantiation and run-time of a service

VNF in the network servers uses a portion of the aforementioned computational and network resources. These metrics are vital for the decision-making process.

A model is introduced to calculate the average slice latency and energy consumption (Figure 48):

$$E_s = \sum_{uv \in \mathcal{E}} \sum_{u_s v_s \in \mathcal{E}_s} E_{uv} z_{uv}^{u_s v_s} + \sum_{m \in \mathcal{M}} \sum_{u_s \in \mathcal{V}_s} E_m z_m^{u_s}.$$

$$D_s = \sum_{uv \in \mathcal{E}} \sum_{u_s v_s \in \mathcal{E}_s} d_{uv} z_{uv}^{u_s v_s} + \sum_{u \in \mathcal{V}} \sum_{u_s v_s \in \mathcal{E}_s} d_u z_u^{u_s v_s} + \sum_{m \in \mathcal{M}} \sum_{u_s \in \mathcal{V}_s} d_m z_m^{u_s}$$

Figure 48: Slice energy and delay model definitions

where e_{uv} represents the energy consumption of an active physical link and e_m the energy consumption of an active server node during one timeslot. Similarly, $d_{uv} d_u$ d_m and uv are the delays introduced due to a traversal of a link u , node m and virtualization middleware m .

In this respect, we formulate the problem of energy-aware low latency SFC management and orchestration as a local long-term constrained optimization task. The key to addressing this challenge is to solve the local multi-objective constrained task, where we jointly seek to minimize latency while reducing energy consumption and guaranteeing sufficient allocated bandwidth.

3.2.1.1 EVALUATION & RESULTS

The performance of the baseline scenarios is normalized to the proposed algorithm entitled SCHE2MA performance, and the plots show the relative gain or loss for each metric. The analysis shows that the performance of the proposed solution in both average energy consumption and average service latency. The energy consumption curves of all figures are normalized based on the SCHE2MA performance to improve legibility. The values are expressed in millijoules (mJ) under the curve of SCHE2MA.

In Figure 49a, we depict the average energy consumption of the examined network of 500 simulations for a varying number of users, normalized based on the SCHE2MA performance (%,mJ). We observe that the energy consumption increases almost linearly with the number of users due to the massive number of transmissions. The reason is that introducing more users to the network generates additional requests that consume more energy during each transmission. Therefore, the overall energy consumption of the network is higher. It is evident that the proposed solution is able to maintain lower energy consumption in all scenarios, reaching almost 17.1% reduction in the case of 100 users. The reason for this behaviour is the ability of SCHE2MA to cluster VNFs into the servers, minimizing the costly communication between servers.

Figure 49b presents the performance of the most critical metric in URLLC services, the average service latency. We observe that the average service latency increases due to insufficient computing resources in servers within the domains as the number of active users grow. However, it has to be noted that SCHE2MA outperforms both baselines by offering a 103.4% reduction in latency for the case of 100 users without increasing the energy consumption, which is a considerable performance improvement while also maintaining lower energy consumption than both baselines. That is possible due to VNF clustering in servers,

which minimizes the number of transmissions in physical media. SCHE2MA demonstrates a clear indication of its ability to conceive better VNF placements that satisfy the latency and energy consumption trade-off.

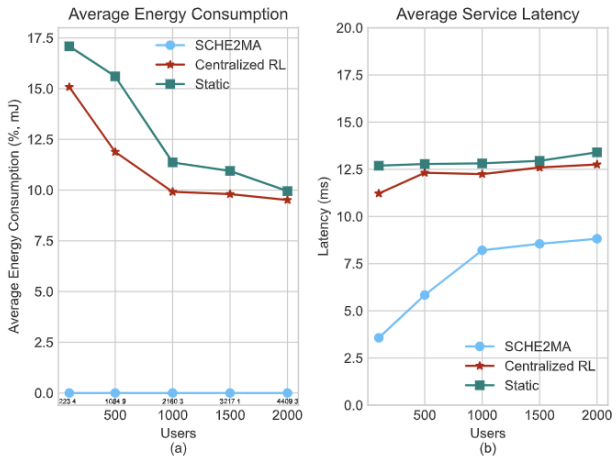


Figure 49: (a) Average energy consumption improvement expressed in percentages, compared to the proposed solution. (b) Average service latency per number of users in multiple traffic scenarios. Lower is better for both figures

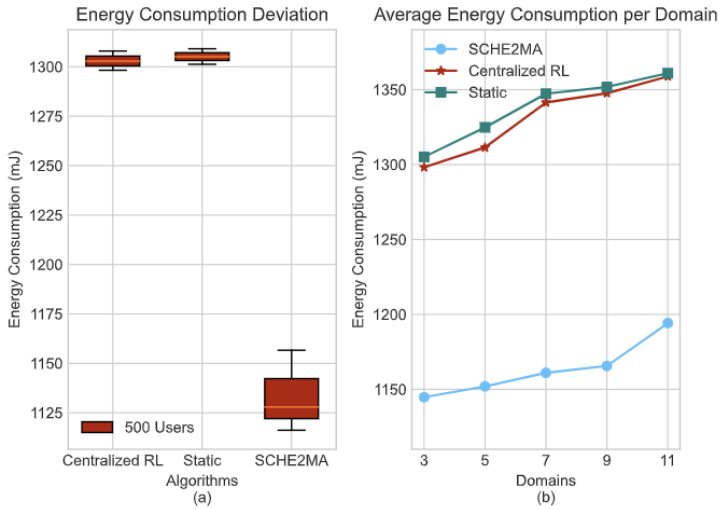


Figure 50: (a) Energy consumption deviation in a 3-domain network with 500 users and 25 SFCs. (b) Average energy consumption in multi-domain network configurations for 500 users and 25 SFCs. Lower is better for both figures.

Figure 50a presents how the energy consumption fluctuates during the operation of each algorithm, specifically for the scenario of 3 domains and 500 users within a simulation cycle. We observe that the maximum difference in energy consumption is 15.91% between the Static solution and SCHE2MA. In Figure 50b, we plot the average energy consumption per domain in order to evaluate the scalability of the given solutions. We observe that the energy consumption steadily increases as we introduce more domains into

the network, hence increasing the number of data that need to be considered when planning a VNF placement. It can be seen that the Centralized RL fails to converge due to the larger state space. SCHE2MA is able to reduce the energy consumption by 14.85% compared to the baseline solutions comparing to scenarios id-est the 9-domain network with 500 users. This behaviour is due to the flexibility and scalability of SCHE2MA's distributed architecture where the decision-making takes place locally in multi-domain agents that communicate through the Auction Mechanism, dividing and sharing that way the immense problem space.

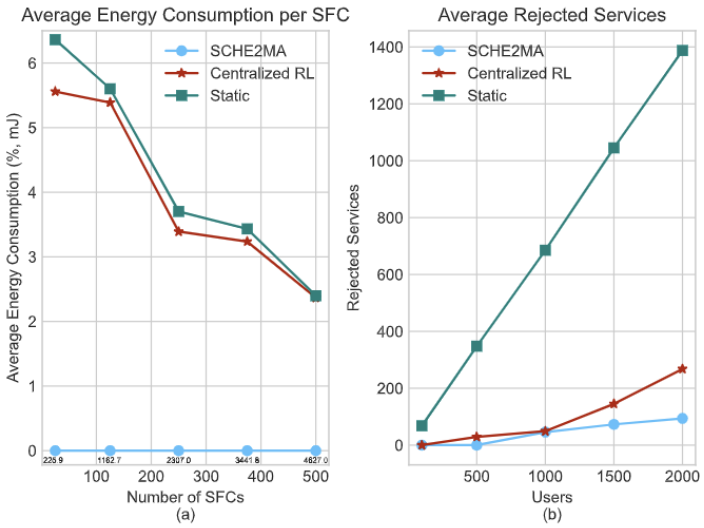


Figure 51: (a) Average energy consumption by the number of SFCs in the network for 500 users. (b) Average number of rejected services for a 3-domain network. Lower is better for both figures.

Figure 51.a outlines the average energy consumption per SFC deployed in the network. We observe that in the case of 25 SFCs, the average energy consumption per SFC of SCHE2MA is reduced by 6.36% compared to the Static solution. Figure 51.b illustrates the average number of rejected services in a 3-domain scenario with a varying number of users. When the number of users increases in a network with finite resources, the number of rejected services increases. Given that the SCHE2MA can re-organize the VNFs, a number of resources can be released. We can conclude that the improvements can be attributed to the VNF consolidation abilities of SCHE2MA.

The results confirm superior performance in multiple scenarios, maintaining the high levels of efficiency in multi-domains scenarios compared to a Centralized RL agent solution.

3.2.1.2 MEASUREMENT METHODOLOGY

The computational resources and networking were modelled with Python 3. An in situ OpenAI Gym environment was developed to interface various RL-based algorithms. The RL agents were developed using TensorFlow and the high-level Keras API open-source library. The simulated environment is a fork of Containernet, an advanced branch of Mininet network emulator. It simulates various features such as, VM hosting, switching, and application code for developing and experimenting with SDN-NFV networks.

The simulations took place in a server cluster with two computers equiped with a 26 Core, 56 Thread Intel Xeon Gold 6230R CPU @ 2.10GHz processor and 256 GB ECC memory.

Figure 52: The simulated topology

The simulated network topology is a variation of the 2005 Nordu European network, from The Internet Topology Zoo, adjusted to accommodate multiple computational domains and fit the requirements of the study. Each node of the topology corresponds to a domain and more nodes are introduced according to the scale of the experiment. The simulation parameters were non-variable during the operation.

3.2.2 STOCHASTIC FEDERATED LEARNING SCENARIO

A novel Statistical Federated Learning (StFL) method [HAT21] [HAT22] has been introduced in the Section 4.2 of MonB5G deliverable D5.3 [D5.3]. The proposed algorithm can learn from non-independent and identically distributed (non-IID) datasets in an offline manner while considering slice-level service level agreement (SLA) long-term statistical constraints. Our approach combines a constrained empirical cumulative distribution function (ECDF) and percentile, both of which are non-convex, to formulate the SLA problem. We introduce a proxy-Lagrangian two-player game strategy to optimize the local FL task while considering both the original SLA constraints and their surrogates. Our StFL method results in a 20% reduction in SLA violations compared to the baseline FedAvg method, and also offers more than x10 overhead reduction and energy efficiency improvement compared to a centralized deep learning approach for SLA-constrained prediction.

In order to ensure further scalability in a B5G/6G massive network slicing, a novel SLA-driven stochastic FL policy has been proposed [SWA22]. Throughout this deliverable, this algorithm will be referred as Policy-based St-FL. At each round of federated learning (FL) process, a subset of active AEs is selected based on their violation rate. Compared to the aforementioned StFL method, policy-based St-FL reduces further the communication overhead and improves the overall energy efficiency in the network.

Section 2.2.1 has presented the most relevant details of the cloud-native implementation of (policy-based and non-policy) St-FL. *This section is focused on the energy computation and provides an energy comparison between the centralized solution and the FL-based algorithms*.

In our implementation of Federated Learning (FL) agents, the FL Server and a module responsible for overall orchestration run together in one Docker container. As introduced in Section 2.2.1, through the REST API, the server and clients can communicate with each other. In our implementation, FastAPI is used as the REST API because it is a modern, open-source, fast, and highly performant Python web framework for building Web APIs with Python 3.6. In particular, a total number of 20 distributed AEs for the federated learning algorithms and for the policy-based StFL only a subset of 15 of these AEs has been considered for the policy-based FL algorithm (i.e., the StFL algorithm). The convergence point for this setup is achieved after 21 rounds in the FL algorithms (see Section 6.1.1.4 of [D3.3] for further details).

3.2.2.1 ENERGY CONSUMPTION COMPUTATION

Based on the TX overhead analysis and the datasets sizes coded in 32 bits, the energy consumption is computed as the sum of two terms [DAL22] [HAT21]: the local computation energy at each CU as well as the energy required for communication over fiber optic transport links.

Following the model introduced in [DAL22], based in the paper presented in [MAO16], the total energy consumption consumed by the network is the sum of the two different terms:

$$E_{total} = E_c + E_t.$$

The first term of the sum, E_c, represents the energy consumed by the use of computational resources. Throughout the document, this term will be referred as computational energy and is defined as follows:

$$E_c = \mu \mathrm{C} \mathrm{D}_s \mathrm{f}^2$$

where μ denotes the effective switched capacitance that depends on the chip architecture [MAO16], C is the number CPU cycles required for computing one bit at each CPU core, D_s denotes the data processed (in bits) and f is the computational capacity of the node in cycles per second.

The second term in the energy formula, E_t, represents the energy consumption needed for the communication process. It is computed as:

$$E_t = \mathrm{p}\mathrm{D}_t/R.$$

where p denotes the transmit power in Watts, R is the data rate in bps and D_t represents amount of data transmitted in bits.

Now let us particularize this formula for the fully centralized SLA-constrained deep learning algorithm (CCL) and the StFL algorithms (both policy-based and non-policy-based). First, let us focus on the energy calculation for the centralized algorithm. It should be noted that in this case the local datasets are transmitted to a central node.

Energy computation for the fully centralized SLA-constrained algorithm (CCL)

$$\mathrm{E}_{\mathrm{t\_CCL}} = p\mathrm{D}_s/R$$

$$E_{c\_CCL} = \mu C \mathrm{D}_s \mathrm{f}^2$$

$$\mathrm{E}_{\mathrm{total\_CCL}} = E_{c\_CCL} + \mathrm{E}_{\mathrm{t\_CCL}}$$

The formula for the (policy-based and non-policy-based) Stochastic FL is described next. Note that in the policy-based StFL a subset of the AEs is selected in each FL round, based on the SLA fulfilment. For the non-policy based or vanilla StFL all the AEs are considered for the Federated Learning process.

Energy computation for (policy and non-policy StFL)

$$D_{s\_sto} = \text{weights} * \text{encoding\_bits} * \text{no\_slices} * \text{no\_clients} * \text{no\_rounds} * \text{no\_features}$$

$$E_{t\_StFL} = pD_{s\_sto}/R$$

$$E_{c\_StFL} = \mu C D_{s\_sto} f^2$$

$$E_{total\_CCL} = E_{c\_StFL} + E_{c\_StFL}$$

Table 13 shows the parameters considered for the computation of the energy consumption.

Parameters	Values
p	$10^{-0.2}$ W
Ds	1875*8*1024 bits
R	1e9 bps
μ	1e-28 F
C	5900/8 cycles/bit
f	2e9 cycles /second
weights	3
number of encoding bits	32 bits/sample
number of slices	3
Number of clients	20 (St-FL), 15 (Policy-FL)
Number of rounds	{21, 50, 60,70,80}
Number of features	3

Table 13: Energy parameters

Table 14 shows the overhead and the energy consumption obtained by the fully centralized SLA-constrained deep learning algorithm (CCL) and the StFL algorithms (both policy-based and non-policy-based). Let us recall that the main difference between the non-policy and policy-based algorithms is that in the non-policy StFL all the AEs are considered in the Federated Learning process, whereas in the policy-based StFL only a subset of the total number of Aes is considered in each round of the FL process. In particular, in this case, 15 clients out of a total number of 20 Aes have been considered for the policy-based StFL. Note that starting from the convergence point of StFL algorithms achieved after 21 rounds in the FL algorithms (see Section 6.1.1.4 of

[D3.3] for further details), more than 10 times overhead and energy consumption reduction are obtained in comparison to centralized approach. Regarding the comparison between the two versions of the StFL. Client subset selection in the policy-based StFL brings a substantial reduction in energy consumption (of about 25%) in comparison to non-policy StFL.

Rounds	21	50	60	70	80
Overhead CCL (KB)	1875				
Overhead non-policy StFL (KB)	44.3	105.5	126.6	147.7	168.8
Overhead policy-based StFL (KB)	33.2	79.1	94.9	110.7	126.6
Energy CCL (J)	4.54				
Energy non-policy StFL (J)	0.11	0.26	0.31	0.36	0.41
Energy policy-based StFL (J)	0.08	0.19	0.23	0.27	0.31
Energy Gain w/non-policy FL	**x 41.27**	**x 17.8**	**x 14.8**	**x 12.7**	**x 11.1**
Energy Gain w/policy FL	**x 56. 32**	**x 23. 65**	**x 19.71**	**x 16.90**	**x 14.78**

Table 14: Overhead and energy comparison between centralized solution and federated learning-based algorithms

Main Findings in WP5 and inputs to WP6 experiments:

Main energy consumption comes from container computational energy because data processing that needs to be done in pods for analytics and decision-making processes. The most critical thing is to reduce data overhead over the network to enhance energy gains (Ec >> Et). Transmission energy could be considered negligible. For this reason, during experiments in WP6, we are considering to relate CPU consumption in the pod level and use relation between CPU load and energy consumption, as formulated in [TAD17].

3.2.2.2 KPI DESCRIPTION AND RESULTS

There are two KPIs that are targeted with this solution.

1. *Improve network energy efficiency by a factor of 10:* Both versions of Statistical Federated Learning (StFL) algorithm (policy-based and non-policy-based algorithms) yield more than x10 energy efficiency gain compared to its centralized SLA-constrained deep learning counterpart, which allows higher scalability and sustainability for analyzing a concurrent number of massive slices. If we compare the non-policy based StFL and the policy-based StFL, the latter achieves a higher energy improvement, because only a subset of the AEs is considered in each round of the FL process, instead of considering all the possible distributed clients.
2. *OPEX reduction due to the automation of service management:* One clear and immediate result of the aforementioned energy efficiency improvement is the OPEX reduction.

4. Conclusions and next steps

In the deliverable, we provided a final report describing the realisation of the MonB5G components that we aim to deploy on the final experimental platforms to show the proof-of-concept scenarios targeted by the project. The scenarios are intended to demonstrate the use of the MonB5G architecture and components for automated network management with two distinctive policies: the optimisation of energy while preserving the promised SLAs, and the security of network services as well as the MonB5G components.

The report presented how WP5 was able to develop building blocks that could be integrated into an external system such as a 5G network, and set up three scenarios to illustrate the innovative features of the MonB5G system.

The first scenario is related to POC1 "Zero-touch multi-domain service management", it consists in placing VNF with the objectives of reducing energy consumption and latency. The results show that the proposed SCHE2MA algorithm is able to make a placement that satisfies the energy consumption and latency balance. In addition, Statistical Federated Learning (StFL) can reduce improve energy efficiency compared to a centralised deep learning approach for prediction under SLA constraints.

The following scenarios are related to POC2, "AI-assisted Security monitoring and enforcement":

- The first scenario is the aLTEr attack is focus on the design and the implementation of the security orchestrator, which is able to mitigate the threat by using the API (Master API of K8S) of Container Service Infrastructure Manager (CISM), to deploy security CNF (CoreDNS to perform DNS over TLS server), and the MS for the monitoring of selected traffic at the interface N6, the AE for the threat detection, and DE to trigger remediation policies. Multiple implementations of different AI/ML algorithms to perform anomaly detection have been propose to compare their performance.
- The second scenario is related to the mMTC attack where a closed-loop control (MS-AE-DE) was implemented to detect and mitigate DDoS attacks on the 5G network control plane more rapidly. In the proposed framework, MS monitors the Attach Requests received by the AMF, the AE uses ML to predict attacks while DE requests AMF to send Registration Reject to suspected devices.
- The third scenario is an implementation and testing of a trust element in the Federated Learning as a security enabler against the poisoned attack: it aims at detecting malicious participants in the federated learning process, and remediating the attack by removing infected models.

The results of the performance tests show that the implemented algorithms allow for rapid data analysis and decision-making time regards to the execution of an action such as VNF lifecycle management or VNF configuration update. This leads to a longer global response time to an event such as a cyber security incident. Nevertheless, the automation of procedures is likely to provide a significant performance gain over any procedure requiring human action.

The next step is to deploy these building blocks, including the MonB5G components and the system on which they operate, and assemble them on the experimental platforms in WP6, to complete the end-to-end performance indicator measurements, and finally to make them ready for the demonstration.

5. References

[D5.3] MonB5G Deliverable D5.3: “Final report on AI-driven MonB5G energy efficiency techniques”.

[D3.3] MonB5G Deliverable D3.3: “Final report of platform integration for the MonB5G AE/MS” .

[DAL22] A. Dalgkitsis et al., "SCHE2MA: Scalable, Energy-Aware, Multidomain Orchestration for Beyond-5G URLLC Services," in IEEE Transactions on Intelligent Transportation Systems, 2022

[HAT21] H. Chergui et al., "Zero-Touch AI-Driven Distributed Management for Energy-Efficient 6G Massive Network Slicing," in IEEE Network, vol. 35, no. 6, pp. 43-49, November/December 2021

[HAT22] H. Chergui, L. Blanco and C. Verikoukis, "Statistical Federated Learning for Beyond 5G SLA-Constrained RAN Slicing," in IEEE Transactions on Wireless Communications, vol. 21, no. 3, pp. 2066-2076, March 2022.

[JAM16] Nicholas A. James, Arun Kejariwal, David S. Matteson, “Leveraging Cloud Data to Mitigate User Experience from ‘Breaking Bad’”, 2016 IEEE International Conference on Big Data (IEEE Big Data), 5-8 December 2016, DOI: 10.1109/BigData.2016.7841013, https://arxiv.org/pdf/1411.7955.pdf

[MMTC] R. Niboucha, S. B. Saad, A. Ksentini and Y. Challal, "Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2022.3230875.

[MAO16] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590-3605, Dec. 2016

[TAD17] S. S. Tadesse, F. Malandrino and C. -F. Chiasserini, "Energy Consumption Measurements in Docker," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017, pp. 272-273