Liability and Trust Analysis Framework for Multi-Actor Dynamic Microservices

Yacine Anser, Chrystel Gaber, Jean-Philippe Wary, Samia Bouzefrane, Méziane Yacoub, Onur Kalinagac, Gürkan Gür

Liability and Trust Analysis Framework for Multi-Actor Dynamic Microservices

Yacine Anser*†, Chrystel Gaber*, Jean-Philippe Wary*, Samia Bouzefrane†, Méziane Yacoub†, Onur Kalinagac‡, Gürkan Gür‡
*Orange, Châtillon, France
†CEDRIC Lab, Cnam, Paris, France
‡Zurich University of Applied Sciences (ZHAW), Switzerland
{name.surname}@orange.com, {name.surname}@cnam.fr, {kalo, gueu}@zhaw.ch

***Abstract*—Microservices architecture has become an increasingly common approach for building complex software systems. With the distributed nature of microservices, multiple actors can contribute to a service, hence affecting the dynamics of the environment and making the management of liabilities and trust more challenging. Service-Level Agreements (SLAs) are critical in that regard and any SLA violation or breach can result in significant financial damages. One major challenge is the lack of indicators to handle the liability and trust in such architectures. To address this issue, in this paper we propose a liability and trust analysis framework, namely the LASM Analysis Service (LAS), for multi-actor dynamic microservices that employs Machine Learning (ML) techniques.**

***Index Terms*—Liability, Trust, Microservices, Machine Learning (ML), Service Level Agreement (SLA)**

I. Introduction

Microservices architecture is increasingly adopted for developing complex cloud-native software systems as it allows greater scalability, flexibility, agility, maintainability and modularity in development and deployment [1]. Combining networking paradigms such as 5G, 6G or cloud-to-edge continuum with microservices-based architectures is beneficial for the resulting applications and networks. This symbiotic relationship is expected to disrupt future IT and network infrastructures as investigated in current research.

However, the use of microservices architecture implies dividing a monolithic functionality into smaller, independent functions potentially provided by multiple actors and orchestrated in a dynamic environment. As a result, the management of liability and trust in microservices architecture is a challenge that needs to be tackled as highlighted by [2] in the context of 5G. Existing works concentrate on Service Level Agreement (SLA) monitoring and Root Cause Analysis (RCA) but there is little literature defining liability metrics that can be used for managing such an infrastructure.

This paper presents a novel contribution called Liability-Aware Security Manager (LASM) Analysis Service denoted as LAS. This framework enhances the functionality of the Liability-Aware Security Manager (LASM) previously presented in [3]. To this end, LAS computes three categories of liability metrics. The first one, Commitment Trust Scores, aims at categorizing the trust that an instance, all instances of a microservices or all microservices of a provider will behave as expected by the commitments taken in SLAs. The second category, Financial Exposure, measures the amount of money that the overall microservice architecture provider might potentially lose with the current composition of microservices. Finally, the third category, Commitment Trends, involves monitoring trends of SLA Violation Rates (SVR) and Instance Commitment Trust in order to predict future violations. After describing the LAS and the metrics it generates, we illustrate and evaluate them with two use cases. Finally, we discuss the results and highlight the strengths of this proposal before concluding our work.

Contributions. This paper describes LASM Analysis Service (LAS) which is a framework designed to compute three categories of liability-related metrics based on the SLAs committed by the components and stakeholders of a microservices architecture as well as their related monitored indicators. The proposed metrics are illustrated with two use cases and evaluated.

Outline. This article is structured as follows. Section II provides an overview of previous work used to build our proposal. Section III describes the proposed LASM Analysis Service (LAS). We then evaluate our contributions through two use cases in Section IV with further discussion in Section V. Section VI reflects on existing related approaches and Section VII concludes the paper.

II. Previous work

The proposed module LAS completes the LASM presented in [2], [3]. The LASM is intended to support an End-to-End (E2E) service providers using a microservice architecture that includes a range of subcomponents. These components may be individual microservices, infrastructure elements, or a mix of both, and they may be supplied by various service providers.

The architecture of the LASM is presented in Figure 1. The LASM Referencing Service (LRS) helps the E2E service provider manage a catalog of microservices and instantiated microservices described using a metamodel called TRAILS

The research leading to these results partly received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no 871808 (5G PPP project INSPIRE-5Gplus). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains.

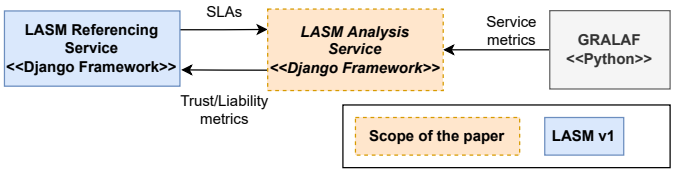


Fig. 1. LASM overall architecture.

for sTakeholder Responsibility, AccountabIity and Liability deScriptor introduced in [3]. The LAS, which is the framework presented in this paper, provides liability-related metrics based on the monitoring of the microservices instances. It receives service metrics from the GRALAF (Graph-Based Liability Analysis Framework) described in [4], a tool that performs near-real time anomaly detection and RCA in a microservice environment. Our proposition integrates with GRALAF, as illustrated in Figure 1. GRALAF monitors various performance metrics of microservices such as service response time and CPU levels. These metrics, alongside SLAs, are used to compute the liability and trust metrics. The SLAs are stored in the TRAILS archive. The LRS is responsible for cataloging the TRAILS associated with each service [3]. The metrics we propose in this article are orthogonal and complementary to those defined in [5] which are calculated by the LRS. These metrics describe a service or a component based on its characteristics while the metrics presented in this paper are based on the observation of the behaviour of the component or service once deployed and in operation.

III. LASM Analysis Service (LAS)

This section describes the internal architecture of the proposed LAS as well as the trust and liability metrics it computes. As described in Figure 2, the LAS uses labelled data sets provided by risk management experts, and the SLAs committed by service providers to generate three categories of metrics, namely Commitment Trust Score, Financial Exposure, and Commitment Trends.

The LAS calculates three types of Commitment Trust Scores, listed as microservice Instance Trust Score (ITS), Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS).

To achieve this, it uses a Multi-Layer Perceptron (MLP) [6] - a type of fully connected feedforward Artificial Neural Network (ANN), and a vector quantification method called k-means. The LAS computes the Financial Exposure to Penalty Risk (FEPR) inspired from financial exposure metric calculated in the field of investments. Finally, two types of Commitments Trends are generated. Using a different type of ANN known as a Self-Organizing Map (SOM) [7], the LAS tracks the changes of the ITS and the SLA Violation Risk over time. This generates two other outputs, namely the Instance Trust Score Trend-Variation (ITS-TV) and the SLA Violation Risk Trend-Variation (SVR-TV).

Further elaboration on these methodologies are provided in the rest of this section.

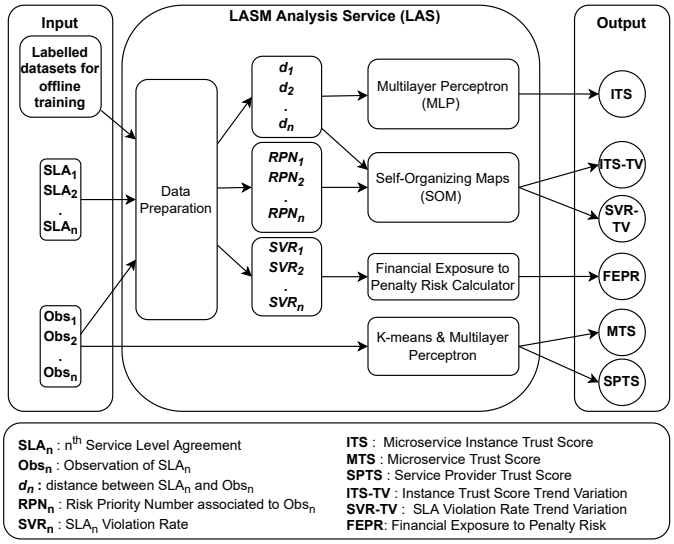


Fig. 2. Overview of LASM Analysis Service (LAS).

A. Data Preparation

The Data Preparation module has two roles. First, it prepares the labelled dataset to train the MLP and the SOM models. Second, it generates the input used by the models in the operational phase based on the SLAs collected from TRAILS data and the observations which correspond to the values of the Service Level Indicators (SLI)[1] collected by GRALAF.

1) Labelling the datasets for training: Each entry in the dataset captures the values of all the SLIs either at regular intervals or upon the occurrence of a SLA violation. These records are annotated with a risk management expert's assessment of the observation, categorized as high, medium, or low.

The MLP requires a labeled dataset which contains a balanced number of normal and abnormal situations. However, we expect that, in most cases, Service Providers and their microservices tend to fulfill their commitments which would result in an imbalanced dataset problem, potentially affecting learning and predictions. Techniques to address this issue include under-sampling the majority class(es), over-sampling the minority class(es), or combining both, with selection dependent on data characteristics. We adopted the `SMOTETomek` algorithm, which is a combination of two algorithms, `SMOTE` [8] and `Tomek-links` [9]. It oversamples the minority class using `SMOTE` and then removes Tomek links to improve the decision boundary between classes.

The training dataset samples are scaled between -1 and 1 using 1 to improve the algorithm's performance:

$$Obs_{scaled} = Obs_{tmp} * (max - min) + min \tag{1}$$

where $min = -1$ and $max = 1$ and:

$$Obs_{tmp} = \frac{(Obs - min(OV))}{(max(OV) - min(OV))} \tag{2}$$

[1]A Service Level Indicator (SLI) is a metric that measures the performance of a service.

Where Obs is an observation and OV is for Observation Vector and max and min are functions that return the maximum value and the minimum value in a vector.

The dataset is split into training and testing sets with a ratio of 75% and 25%. It underwent numerous random splits, a process repeated until the test set reliably reflected behaviors that had not been observed before. This helps evaluate the model's performance more robustly and ensures that it generalizes well to new data, not just the data it was trained on. Finally, we use GridSearchCV [10] to set the hyperparameters of the MLP, including the number of hidden layers, the number of nodes in each layer, the activation function, the learning rate, and the solver. This method does an exhaustive search over specified parameter values for the MLP. As recommended by the methodology, we first define our GridSearchCV strategy by specifying the expected scores, then we determine the cross-validation splitting strategy as Time Series Split.

2) Distance between committed SLA and observation: The data preparation module produces a vector D, consisting of a series of distances d, which measures the degree of compliance with the SLA. The computation of these distances depends on the specific characteristics of the SLA. For example, the following distance is suitable for an SLA which penalizes under-performance:

$$d_i = \frac{SLA_i - Obs_i}{max(Obs_i)} \tag{3}$$

i is a unique pairing, with SLA_i as the value for the i^{th} SLA and Obs_i as the corresponding i^{th} observation value.

3) Severity of deviation between committed SLA and observations: Severity is measured on a scale ranging from 0 to $NCat$, indicating the severity level of the distance between the committed SLA_i and the related observation Obs_i.

$$s_i = \begin{cases} 0, & R_0(SLA_i, Obs_i) \\ 1, & R_1(SLA_i, Obs_i) \\ \dots & \\ NCat, & R_{NCat}(SLA_i, Obs_i) \end{cases} \tag{4}$$

s_i depends on the relationship between SLA_i and Obs_i as determined by various relational conditions R_0,R_1,...,R_{NCat}. Each condition $R(SLA_i, Obs_i)$ involves comparing SLA_i and Obs_i using relational operators.

4) SLA Violation Rate: The Data Preparation module computes the SVR for each SLA_i as follows:

$$SVR_{i,l} = \frac{1}{TN} \sum_{t=0}^{TN} f(s_{i,t}, l) \tag{5}$$

l ranges from severity 0 to $NCat$. SVR is calculated for each l value. TN represents an observation time frame, while the function f quantifies severity occurrences within time range TN. It is defined as follows:

$$\mathbf{f(x, l)} = \begin{cases} 1, & x = l \\ 0, & x \neq l \end{cases} \tag{6}$$

5) Risk Priority Number: The data preparation module computes a Risk Priority Number (RPN) for each observation i and for each SLA_i based on the severity of deviation s_i:

$$RPN_i = s_i * SVR_{i,l} \tag{7}$$

B. Instance Trust Score (ITS)

Our objective is to classify each observation acquired by GRALAF. We explored multiple neural network models to perform such multi-classification tasks, such as MLP and Support Vector Machines (SVM), and the results obtained with the MLP were the most satisfactory.

The input of the MLP is the distance vector D detailed in Section III-A2. The number of nodes in the input layer corresponds to n, which is the size of the vector D. The number of nodes in the output layer corresponds to the number of class of trusts we define. In our case, there are three classes of trust, namely High Level of Trust, Medium Level of Trust and Low Level of Trust. We also use the function `Softmax` in the output layer. To adapt to the dynamic and ever-changing nature of our environment, we train and test our model offline and deploy it in the production environment. Periodically, we retrain the model with newly labeled and validated data, followed by an evaluation of its performance using metrics including accuracy, precision, recall, F1-score, and the confusion matrix. Furthermore, we conducted Receiver Operating Characteristic (ROC) curve analysis using the one-vs-rest approach, specifically targeting the "High level of trust" class. The Area Under Curve (AUC) score was also calculated as part of this assessment. Furthermore, if issues like drift are detected during monitoring, we may initiate the model's retraining process.

C. Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS)

The aim is to determine the level of trust in a microservice and the providers of that service from multiple observations of the service instance at a specific time point, the level of trust in the microservice and the providers of the service. (Note that one provider can offer multiple classes, and one class may involve several providers). The method for computing the MTS involves using the k-means algorithm, a Vector Quantization (VQ) technique. This algorithm is a popular clustering technique due to its simplicity and ability to scale large data sets. We have opted to use k-means algorithm because it is relatively easy to implement and is applicable to numeric and continuous data.

The k-means algorithm is used to perform VQ on several observations of a commitment on the same instance of a microservice. Let n be the number of instances of a microservice and o_i be the observation of SLA_i. At an instant T, we measure the observation o for the n microservice instances. These measurements form the observation vector O. Then we represent the observations by a prototype ($centroid$) using k-means. The vector O and the number of clusters k are the algorithm's parameters. To determine k, we use the Elbow Method [11], which assists in determining the ideal number of clusters in datasets. This method plots cluster numbers against a performance metric, pinpointing the optimal cluster count where further additions do not notably enhance the model. k-means gives the codebook as output. Using the codebook, we map the code to $centroid$ in order to obtain the prototype observation. The prototype observation and the commitment

SLA_i are processed to obtain the distance d. This process is repeated for all commitments made on the microservice, and the resulting vector D is presented to the MLP model to obtain the trust class.

For the SPTS, we need to make some modifications to the existing methodology. Specifically, instead of inputting the observations into the k-means algorithm, we input the MTS of the relevant service provider's microservice. These values are encoded using one-hot encoding techniques [12]. The resulting prototype generated by the k-means algorithm represents the SPTS. The Figure 3 outlines the process of calculating the two metrics MTS and SPTS.

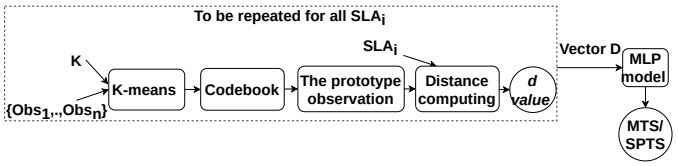


Fig. 3. MTS and SPTS computation process.

D. Temporal Evolution on the Self-Organized Map of the ITS and the SLA Violation Risk

Our objective is to perform a real-time observation of the ITS and the SLA Violation Risk to have an efficient tracking of the metric dynamics. Also, we want to determine when the metrics are entering a non-desired state represented by a forbidden area and a warning area on a map. For that purpose, we use a special class of ANN called Self-Organizing Maps (SOM). The following gives an overview on how it is applied.

We create two SOMs for each service class, one for the TSI metric (map_1) and another for the SLA Violation Risk metric (map_2), using the same process for both. This involves a training phase (Steps 1-3) and an operation phase (Step 4), implemented using minisom [13], a Python tool for SOM training and construction.

Step 1. The data acquired from measurements are transformed into input data for the SOM map. Specifically, we compute the d vector (the input for the map_1) and the RPN vector (the input for the map_2).

Step 2. The SOM is trained with all available data. The parameters for training, including the map's dimension, learning rate, and neighborhood coefficient, are chosen based on empirical benchmarks. The quality of the resulting mapping is assessed using metrics including the quantization error, the topographic error, the silhouette score, the distortion and the neighborhood preservation [14].

Step 3. After the training phase, a label is assigned to each neuron in the grid. This label corresponds to a particular class, determined by analyzing the u-matrix and the component planes representation. Two types of area are defined: the "forbidden" area, which corresponds to neurons being labeled "Low level of trust" and "High level of risk" for map_1 and map_2, respectively and the "warning" area which corresponds to the to neurons being labeled "Medium level of trust" and "Medium level of risk" for map_1 and map_2, respectively.

Step 4. During the operational phase, the inputs are projected onto the map, the sequence of nodes in time that forms a trajectory on the map depicting the movement of the metrics. A detailed alert message is triggered if an input is projected in a restricted area.

E. Financial Exposure to Penalty Risk (FEPR)

Financial Exposure to Penalty Risk (FEPR) is a term that comes from the financial and risk management world. It is used to measure the amount of money that an investor might lose on an investment. In our context, we use it to quantify the financial risk a microservice architecture provider integrating multiple microservice components is exposed to when it offers a service to a customer. It is defined as

$$FEPR_{i,j} = \sum_{i=0}^{3} SVR_{i,j,l} * (rew_l - pen_l) \tag{8}$$

where rew corresponds to the reward that the microservice architecture provider earns if it honours its commitments, and pen the penalty the microservice architecture provider must repay if it does not meet its commitments. The SVR is the SLA Violation Rate explained in the subsection III-A4.

IV. Evaluation and Result

To evaluate our contribution, we showcase a practical application of the LAS through two use cases. For both, we defined several scenarios to highlight the characteristics of our contribution. The LAS was deployed on a Kubernetes container platform with the help of Python libraries such as Numpy, Pandas, Scikit-Learn and Matplotlib. In the following, we describe the use cases and present the results obtained.

A. Use case n°1 - PacketFabric SLA

1) Use Case Description: In the first use case, we work with synthetic data that we generated based on the SLA of PacketFabric [15]. We assume that PacketFabric provides a service that consists of deployment and management of network services. This use case also illustrates that the LAS is applicable in scenarios beyond microservices.

Service Level Agreement PacketFabric commits to the following service level metrics which are denoted as SLA_i with its related observation $\mathcal{O}_i$ [15]:

- Network availability: Deliver availability of at least 99.988% in the network → SLA: SLA_0, related observation: O_0.
- Latency: Deliver a network service with an end-to-end latency lower than 95ms → SLA: SLA_1, related observation: O_1.
- Packet loss: Deliver a network service with a network packet loss across the network lower than 0.14% → SLA: SLA_2, related observation: O_2.

The *Core Network Availability*, *Latency Metric Extended* and *Loss Metric Exceeded* tables provided in [15] summarize the levels of SLA penalties and the corresponding penalties that the service provider is eligible to receive if SLA_0, SLA_1 and SLA_2 are not met, respectively. The SP deploys the LAS in order to evaluate the network service using the available metrics presented in Section III.

2) Dataset for training phase: For this use case, we used synthetic data to overcome the lack of real-world data.

For dataset creation, we relied on the SLA structure from PacketFabric as our starting point. Utilizing this information, we crafted a distribution function, employing both a Gaussian Mixture Distribution (GMD) and a uniform distribution to mimic the SLA attributes. Consequently, our dataset mirrors the SLA specifications of the packetFabric service. The methodology used consists of three steps. The first step involves generating five datasets with five GMDs with the same mean but different variances. The second step involves drawing a uniform number of samples from these five datasets to create the final dataset. Finally, that final dataset is timestamped.

We illustrate this methodology for the Core Network Availability service level. The first step will be to generate the five datasets. The GMD takes the following form:

$$X \sim 0.90 * \mathcal{N}(\mu_1, \sigma) + 0.04 * \mathcal{N}(\mu_2, \sigma) + 0.03 * \mathcal{N}(\mu_3, \sigma) + 0.01 * \mathcal{N}(\mu_4, \sigma) + 0.005 * \mathcal{N}(\mu_5, \sigma) + 0.005 * \mathcal{N}(\mu_6, \sigma) \quad (9)$$

The means μ_1 through μ_6 from the Core Network Availability table are mean values for each interval, calculated as μ_1 = 0.9999, μ_2 = 0.9995, μ_3 = 0.99673, μ_4 = 0.991, μ_5=0.986, and μ_6 = 0.961, identical across all five GMDs.

The standard deviation σ values differ for each dataset. Specifically, for dataset n°1, $\sigma_{[1,2,3,4,5,6]}$ = 0.001, for dataset n°2, $\sigma_{[1,2,3,4,5,6]}$ = 0.005, for dataset n°3, $\sigma_{[1,2,3,4,5,6]}$ = 0.01, for dataset n°4, $\sigma_{[1,2,3,4,5,6]}$ = 0.05, and for dataset n°5, $\sigma_{[1,2,3,4,5,6]}$ = 0.1.

To create the final dataset, we draw samples from the five datasets using a uniform distribution and timestamp the resulting synthetic time-series dataset. This process is repeated for the other two service levels, resulting in a final synthetic dataset of 4230 samples.

3) Datasets for operational phase: Six datasets were created to evaluate our metrics. The first dataset is used in our evaluation of ITS, ITS-TV, and SVR-TV. The other five datasets were created to evaluate MTS and SPTS. They represent data generated by five instances of two different microservices provided by a unique Service Provider.

The first dataset is illustrated in first three rows of Table I for SLA_0, SLA_1 and SLA_2, respectively. The dataset spans over six Time Windows (TW), each having a different scope.

TABLE I
TIME WINDOWS AND CORRESPONDING ITS (COMPLY: AS EXPECTED, ↓ : LESS THAN EXPECTED, ↑ : HIGHER THAN EXPECTED)

	TW0	TW1	TW2	TW3	TW4	TW5
$\mathcal{O}_0$	Comply	0.0048% ↓	0.0048% ↓	0.01288% ↓	4.1% ↓	Comply
$\mathcal{O}_1$	Comply	Comply	63% ↑	Comply	69% ↑	Comply
$\mathcal{O}_2$	Comply	Comply	Comply	Comply	43% ↑	Comply
ITS	High	Medium	Low	Low	Low	High

Nine case studies are included in the last five datasets, and they are all denoted using the notation $O_{x,y,z}$, where x denotes the observation index, y the instance number and z the microservice number, respectively. The following presents these nine case studies:

- **CS1** : All the observations behave as expected.
- **CS2** : $O_{1,1,1}$ results in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS3** : $O_{2,1,1}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS4** : $O_{2,1,1}$ results in 10% of MRC penalty and $O_{3,2,1}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS5** : $O_{1,1,1}$ results in 10% of MRC penalty, $O_{2,2,1}$ result in 10% of MRC penalty and $O_{3,3,1}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS6** : $O_{1,1,1}$ results in 20% of MRC penalty and $O_{1,1,2}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS7** : $O_{2,2,1}$ results in 20% of MRC penalty, $O_{2,3,1}$ result in 20% of MRC penalty and $O_{2,1,2}$ result in 30% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS8** : $O_{3,2,1}$ results in 20% of MRC penalty, $O_{2,3,1}$ result in 30% of MRC penalty and $O_{3,1,2}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.

4) Results:

a) Instance Trust Score (ITS):

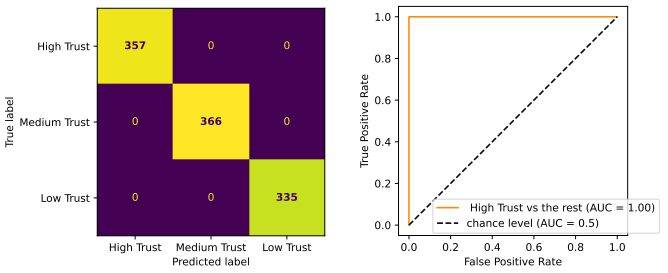
Offline phase results First, the dataset for the training phase is labeled, resampled, scaled, and split into training and testing sets. The hyperparameters determined by the *GridSearchCV* algorithm are the learning rate of 0.0001, a single hidden layer comprising 15 neurons, and the hyperbolic tangent function as the activation function. We then start the evaluation with the confusion matrix (Figure 4(a)). As we can observe, all the samples have been accurately classified. Then, we evaluate the model using the metrics defined in Section III-B and show the results in Table II. The classifier performs with very high precision and recall scores for all three classes, along with high specificity and F1-scores. Additionally, the geometric means for all three classes are also high, indicating that the classifier is unbiased towards any particular class.

TABLE II
CLASSIFICATION RESULTS.

	Precision	Recall	Specificity	F1-score	Geo. mean
High Trust	1.00	1.00	1.00	1.00	1.00
Medium Trust	0.99	1.00	1.00	1.00	1.00
Low Trust	1.00	0.99	1.00	1.00	1.00

We then generated the ROC curve (Figure 4(b)) and calculated the AUC. An AUC of 1 shows that the model accurately identifies instances of the target class.

Operational phase results. The goal is to assess whether the ITS progress for each TW aligns with the anticipated trend of the three observations over time. The results displayed in



(a) Confusion matrix

(b) One-vs-Rest ROC curves: High Trust vs (Medium Trust & Low Trust)

Fig. 4. Confusion matrix and ROC curve.

the fourth row of Table I show that the LAS correctly evaluates the ITS over time. Indeed, the ITS is *High* at **TW0** and **TW5**. This fits with the observations as the three SLAs are met. At **TW1**, the LAS indicates that the ITS is *Medium*. This is accurate because it is clear that during this TW the SLA_1 and the SLA_2 are met but the $\mathcal{O}_0$ deviate slightly from the committed value SLA_0 which is enough to bring down the ITS to *Medium* as it is built as such. During **TW2**, **TW3** and **TW4**, the ITS is *Low*. This can be explained as at **TW2** the $\mathcal{O}_0$ deviates slightly and $\mathcal{O}_1$ deviates moderately which brings the ITS down to *Low*. At **TW3** we can observe that $\mathcal{O}_0$ deviates totally and at **TW4** the three observations totally deviate.

b) Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS): Table III summarizes the MTS and SPTS for this use case in operational phase. In CS1, all observations behave as expected, resulting in a *High* rating for CTS 1, CTS 2, and SPTS. In CS2, CTS 1 is rated *Medium* due to a slight deviation in class instance availability which is enough according to the *Core Network Availability* table. However, CTS 2 is *High* resulting in a *High* rating for SPTS. In CS3, all observations in CTS 1, CTS 2, and SPTS are *High* with only minor latency deviations in the first class. In CS4, CTS 1 is *Medium* due to slightly high latency in the first instance and packet loss in the second instance. However, CTS 2 is *High* resulting in a *High* rating for SPTS. In CS5, CTS 1 is rated *Low* due to three deviating observations: availability in the first instance, latency in the second instance, and packet loss in the third instance. CTS 2 is *High* resulting in a *Medium* rating for SPTS. In CS6, CTS 1 and 2 are rated *Medium* due to a slight deviation in availability for the first instance of each class, resulting in a *Medium* rating for SPTS. In CS7, CTS 1 is rated *Low* due to deviating latency in the second and third instances. CTS 2 is rated *Medium* due to excessively high latency in the first instance, resulting in a *Medium* rating for SPTS. In CS8, CTS 1 is rated *Medium* due to a slight deviation in error rate and in the latency for the second and third instance, CST 2 is *High* because the the deviation in latency is quite minor, resulting in a *High* SPTS.

c) Trend Variations of ITS and SVR: For both of our maps, we set the learning rate to 0.0001 to ensure a stable SOM map. Neuron weights were randomly initialized, and the Gaussian function was chosen as the neighborhood function due to its common usage. We chose to create a rectangular map with dimensions of 15 by 15 for the ITS-TV map and 10 by 10 for the SVR-TV map.

TABLE III
MTS AND SPTS (L: LOW, M: MEDIUM, H: HIGH).

	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8
CTS-1	H	M	H	M	L	M	L	M
CTS-2	H	H	H	H	H	M	M	H
SPTS	H	H	H	H	M	M	M	H

TABLE IV
SOM - EVALUATION.

Metrics	ITS-TV map	SVR-TV map
Quantization error	0.03	0.02
Topographic error	0.08	0.03
Silhouette score	0.65	0.78
Distortion	0.71	0.65
Neighborhood preservation	0.86	0.75

Table IV presents the evaluation results of two SOM maps. The low quantization and topographic errors (0.03/0.08 for ITS-TV map, 0.02/0.03 for SVR-TV map) indicate that the SOM maintained the spatial and topological relationships of input data points accurately. The high silhouette scores (0.65 for the ITS-TV map, 0.78 for the SVR-TV map) show excellent clustering quality, and the low distortion values (0.71 for the ITS-TV map, 0.65 for the SVR-TV map) suggest tightly packed data points in each cluster. The high neighborhood preservation score (0.86 for ITS-TV map, 0.75 for SVR-TV map) demonstrates effective preservation of spatial relationships between neighboring data points.

Interpretation of the ITS-TV Map. After an initial analysis of the u-matrix and the components plane, we have arrived at a comprehensive interpretation which is presented in detail in Figure 5. Our analysis has led us to identify a total of six distinct areas on the map, which can be broadly categorized into three forbidden areas and three warning areas. Each of these areas is associated with a specific color code and a detailed description of the anomaly observed in that area. The uncolored neurons indicate an area where there are no observations associated with them. The normal area where no anomaly is detected is in green. The first forbidden area, in red, corresponds to cases where the SLA_0 is 3% above the committed value. In the second forbidden area, in blue, the SLA_2 exceeds the commitment by 26%. In the third forbidden area, in black, the SLA_1 is 32% higher than the committed value. In the first warning area, in orange, the SLA_1 is 14% higher than the committed value. In the second warning area, in yellow, the SLA_0 is 1% higher than the committed value. Finally, the third warning area, in cyan, the SLA_2 is 16% above the committed value.

Operational Phase for the ITS-TV Map. To illustrate how the map can be practically employed, we present the following scenario: the service begins functioning as expected, the input data is then projected in the green area. However, the latency rises from 94ms to 95.5ms. The input data is then projected in

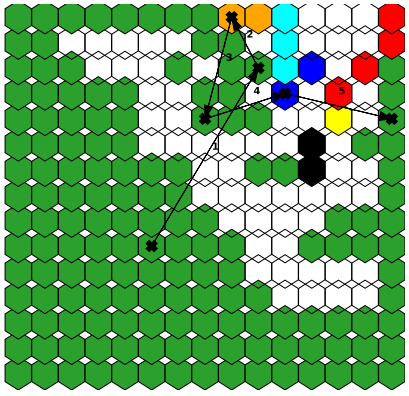


Fig. 5. ITS-TV Map with data insight.

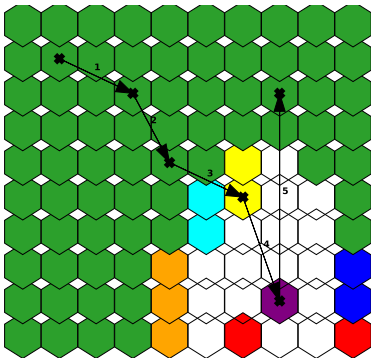


Fig. 6. SVR-TV Map with data insight.

the first warning area which raises an alert. Then, the service returns to its normal state. Next, the packet loss rate increases to reach 0.2% and the input data is projected into the third forbidden area. Eventually, the Network Service returns to its normal behavior which brings the input data into the green zone. The path created by the various inputs is visualized Figure 5.

Interpretation of the SVR-TV Map. Figure 6 presents an overview of the distribution of the training dataset on the SOM map. Our analysis has led us to identify three forbidden areas and three warning areas. As before, the uncolored neurons mean no associated observations in that area. The normal area is in green. The first forbidden area, in blue, corresponds to a probability of over 58% that SLA_1 will be violated. The second forbidden area, in purple, corresponds to a probability of over 55% that SLA_2 will be violated. The third forbidden area, in red, corresponds to a probability of over 60% that SLA_0 will be violated. The first warning area, in orange, corresponds to a probability of over 27% that SLA_0 will be violated. The second warning area, in cyan, corresponds to a probability of about 13% that SLA_1 will be violated. Finally, the third warning area, in yellow, corresponds to a probability of about 18% that SLA_2 will be violated.

Operational Phase for the SVR-TV Map. To demonstrate the practical application of the map, we consider the following scenario: once the service is up and running as anticipated, the input data is then mapped onto the green region. However, the packet loss rate starts to increase slightly; the input data is then projected into the third warning area, and an alert is raised. The packet loss rate continues to increase, and in turn the risk that this SLA will be violated increases, so the data is projected into the second forbidden area, an alert indicating that the high probability of this SLA being violated is raised. Finally, the network service resumes its normal behavior, resulting in the input data being brought back into the green zone.

d) Financial Exposure to Penalty Risk:

Highlight. In the following, we use the FEPR metric to analyze the PacketFabric use case and examine the relationship between the increased risk of SLA failure and the corresponding FEPR. For that purpose, we apply penalties for availability, latency, and packet loss rate, as defined in [15], using the same scenario used previously to demonstrate the variation in SLA Violation Trend.

Result. At the start of the monitoring period, the FEPR metric is at 0$, indicating that the service is operating as expected. However, as time goes on, there is a gradual degradation in packet loss rate, which increases the risk of SLA violations and causes the FEPR to decline to -200$ and eventually to -1200$. The situation worsens, further raising the risk of SLA violations and resulting in a sharp increase in the FEPR to -2200$. The service eventually returns to normal operation, reducing the risk of SLA violations and causing the FEPR to go back to 0$.

B. Use case n°2 - EdgeX SLA:

1) Use Case Description: The testbed for the second use case is illustrated in Figure 7. The EdgeX is used for IoT device management and is an open source software framework that offers device and application interoperability at the IoT edge. Apart from the main microservices from EdgeX, we also deployed some additional services. `MQTT-broker` service serves as an intermediary between the IoT devices supporting MQTT and the `MQTT-device` service introduced by us. We deployed multiple instances of a microservice application that emulates IoT devices, sends random sensor data periodically and can be controlled via MQTT. To send the received sensor data to an external server which hosts Fledge, a data exporter service called `exporter-fledge` is also deployed in the EdgeX environment. Fledge is an open source framework and community focused on IoT devices for the industrial edge. Locust is an open source performance testing tool capable of simulating a large number of concurrent users and it is used to generate traffic on the Edgex ecosystem. GRALAF [16] periodically queries Prometheus, a widely-used open-source system for gathering, storing, and querying metrics. Prometheus is configured to scrape the metrics every 60 seconds and to store them in a time-series database that is organized by SLA name.

For the infrastructure setup, we use five virtual machines running on an OpenStack cloud infrastructure. A MicroK8s based Kubernetes cluster, which hosts Edgex services, Locust and all other required system services like Prometheus and Istio are deployed using three of the VMs. These three virtual machines have 4 vCPUs, 8 GB of RAM, and 160 GB of SSD storage. Istio provides traffic related metrics such as response

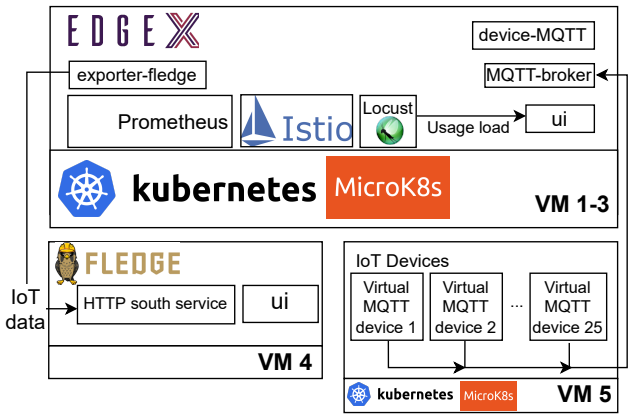


Fig. 7. Test environment for Use Case 2 VM indicates in which virtual machine the services are deployed.

time and error rate while Prometheus scrapes all the metrics from the available providers like Kubernetes infrastructure service and Istio. A Fledge server is hosted by one VM, and 25 MQTT-based virtual IoT device applications are deployed in a MicroK8s environment on the other VM. These two virtual machines have the following resource assignments: 1 vCPU, 2GB of RAM, and 120GB of SSD storage. The service-VM mapping for this use case is illustrated in Figure 7.

Target Service. EdgeX service is divided into four services, specifically the *core*, *supporting*, *system management* and *devices* services. Each service is composed of one or several microservices. Each service or microservice is provided by a service/microservice provider. For the evaluation, we focused on the core service, namely the `core-metadata` microservice. It communicates with other microservices such as `core-command`, `UI` and `device-mqtt`.

Service Level Agreement. An SLA is established between the parties where the provider of the `core-metadata` microservice committed to the following service level metrics:

- Service Availability: Deliver availability of at least 99% for the service → SLA: SLA'_0, related observation: O'_0.
- Service Latency: Deliver a service with a latency lower than 100ms → SLA: SLA'_1, related observation: O'_1.
- Service Error Rate: Deliver a service with an error rate lower than 0.5 → SLA: SLA'_2, related observation: O'_2.

Table V lists the penalty charges the consumer of the core microservice is entitled to receive if the commitment is not met with a Monthly Recurring Charge (MRC) of 10000$.

2) Dataset for the training phase: For this use case, we adopted Chaos Mesh to perform realistic perturbations to create a representative dataset. It is a platform that orchestrates faults in Kubernetes. The experiment's workflow involves multiple steps. Initially, the service's steady state is established, representing its standard operation while fulfilling all SLAs. Next, real-world events are defined that can disrupt the service. Chaos Mesh is tightly integrated with Kubernetes and provides various potential failures in Kubernetes clusters. For our assessment, we mainly utilized network outages, memory and CPU stress, latency injection, and pod termination events, all of which can lead to breached SLAs. We end up with a dataset of 9718 samples.

TABLE V
SLA PENALTIES FOR EDGEX: AVAILABILITY, LATENCY, AND ERROR RATE

Availability	Penalty	Latency	Penalty	Error rate	Penalty
>=99.862% <99.988%	10% of MRC	10% above SLA	10% of MRC	10% above SLA	10% of MRC
>=99.445% <99.862%	20% of MRC	20% above SLA	20% of MRC	25% above SLA	25% of MRC
>=98.889% <99.445%	30% of MRC	40% above SLA	30% of MRC	50% above SLA	50% of MRC
>=98.334% <98.889%	40% of MRC	60% above SLA	40% of MRC	75% above SLA	40% of MRC
>=96.667% <98.334%	60% of MRC	75% above SLA	50% of MRC	100% above SLA	60% of MRC
<96.667%	100% of MRC	100% above SLA	60% of MRC		

3) Datasets for the operation phase: Four datasets were created to evaluate our metrics. The first dataset is used in our evaluation of ITS, ITS-TV, SVR-TV. The last three datasets were created to evaluate MTS and SPTS. They represent data of three instances of one microservice provided by a unique service provider. The first dataset is illustrated in the first three rows of Table VI for SLA'_0, SLA'_1, and SLA'_2, respectively. It spans over six TW, each having a different scope.

TABLE VI
TIME WINDOWS AND CORRESPONDING ITS (COMPLY : AS EXPECTED, ↓ : LESS THAN EXPECTED, ↑ : HIGHER THAN EXPECTED)

	TW0	TW1	TW2	TW3	TW4	TW5
$\mathcal{O}_0$	Comply	Comply	0.0053% ↓	Comply	0.0053% ↓	Comply
$\mathcal{O}_1$	Comply	13% ↑	Comply	48% ↑	Comply	10% ↑
$\mathcal{O}_2$	Comply	Comply	Comply	Comply	13% ↑	9% ↑
ITS	High	Medium	Low	Low	Low	Low

The last three datasets contain a total of nine case studies, each designated with the notation $O_{x,y}$ where x represents the observation index and y indicates the microservice number. Below are the details of these nine case studies:

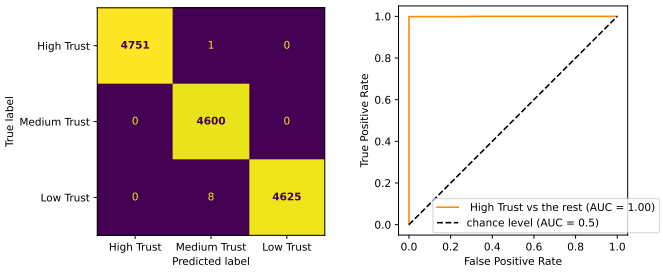
- **CS0**: All the observations behave as expected.
- **CS1**: $O_{0,1}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS2**: $O_{1,1}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS3**: Each observation $O_{0,1}$ and $O_{1,1}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS4**: Each observation $O_{0,1}$ and $O_{1,1}$ results in 10% of MRC penalty. $O_{2,1}$ result in 25% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS5**: Each observation $O_{0,1}$ and $O_{0,2}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.

- **CS6**: Each observation $O_{0,1}$, $O_{0,2}$ and $O_{1,1}$ results in 10 % of MRC penalty. $O_{1,2}$ results in 25% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS7**: Each observation $O_{0,1}$, $O_{0,2}$ and $O_{1,1}$ results in 10% of MRC penalty. Additionally, $O_{1,2}$, $O_{2,1}$ and $O_{2,2}$ results in 20% of MRC penalty each. The remaining observations conform to the expected behavior.
- **CS8**: Each observation $O_{1,1}$, $O_{1,2}$ and $O_{1,3}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.

C. Results

a) Instance Trust Score:

Offline phase results. Prometheus may produce raw data that contains missing values. To handle this, we use the Multivariate Imputer method outlined in [17]. We labeled, resampled, scaled, and split the training dataset. The GridSearchCV procedure yielded identical values for the learning rate and activation function. However, some differences were noted in the values of other parameters such as the number of hidden layers (two instead of one), and the size of the hidden layers (30 instead of 15). We evaluate the model on the testing dataset. The confusion matrix is presented in Figure 8-a. We can see that only nine samples were misclassified (eight as medium trust instead of low trust, and one as medium trust instead of high trust), which is acceptable considering the total number of samples. From the confusion matrix, we calculate the evaluation metrics presented in Table VII.



(a) Confusion matrix

(b) One-vs-Rest ROC curves: Trust vs (Medium Trust & Low Trust)

Fig. 8. Confusion matrix and the ROC curve.

TABLE VII
CLASSIFICATION RESULTS.

	Precision	Recall	Specificity	F1-score	Geo.mean
High Trust	1.00	1.00	1.00	1.00	1.00
Medium Trust	1.00	0.98	1.00	0.99	0.99
Low Trust	0.98	1.00	1.00	1.00	1.00

The classifier's performance exhibits a precision of 1.00 for the first class, 1.00 for the second class, and 0.99 for the last class, indicating a low false positive rate. Moreover, the model displays a very high accuracy in identifying positive cases for all three classes, as evidenced by a recall of 1.00 for the first class, 0.98 for the second class, and 1.00 for the last class. Additionally, the specificity is high for all three classes (1.00), signifying that the classifier excels at identifying negative cases. The F1-scores for all three classes are high (1.00, 0.99, and 0.99), indicating that the classifier can accurately identify most positive cases while minimizing false positives. Furthermore, a high F1-score suggests that the classifier is effectively balancing the trade-off between the three classes. Similarly, the geometric means are high for all three classes (1.00, 0.99, and 1.00), indicating that the classifier can correctly identify both positive and negative cases without showing bias towards any particular class.

Finally, we generated ROC curve (Figure 8-b) and calculated AUC, with a result value of 1, indicating the model's precise identification of target class instances.

Operational phase results. To assess the performance of the offline model for online classification, the results presented in the last row of Table VI demonstrate that the LAS is successful in accurately assessing the ITS over time. Indeed, at **TW0**, the ITS is classified as *High* since all three commitments were met during that period, aligning with our expectations. However, at **TW1**, the ITS is deemed *Medium* due to the unexpected behavior of O'_1, resulting in a decrease in the ITS. This deviation is not considered critical, and the Latency is not significant based on the criteria outlined in Table V. Following this disruption, the ITS gradually improves as all three commitments are met. At **TW2**, the ITS drops to *Low* since even a minor variation in the availability of O'_0 is deemed crucial according to the delimitation provided in Table V. From that moment, the ITS will remain low; indeed, at **TW3**, the ITS is **Low** since the latency is far too high compared to the commitment. At **TW4**, the ITS is *Low* since both O'_0 and O'_1 deviate from their expected behavior. At **TW5**, the ITS falls to *Low* as both O'_1 and O'_2 deviate slightly from their anticipated behavior. Finally, the ITS returns to *High* as all three observations behave as expected.

b) Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS): We report the results in Table VIII. Since the provider only offers one class in this particular case, the score of the provider is equivalent to the score of the class, i.e., CTS. We observed that for CS1, all observations behaved as expected, leading to a CTS rating of *High*. However, for CS2, the availability of the microservice on cluster0 slightly deviated, resulting in a CTS rating of *Medium*. Similarly, for CS3, the availability and latency did not behave as expected on cluster0, resulting in a CTS rating of *Low*. For CS4, the three commitments on cluster0 were not met, leading to a CTS rating of *Low*, indicating that the issue was with the cluster0 and not the microservice. For CS5, the microservice availability on both cluster0 and cluster1 slightly deviated, leading to a CTS rating of *Low*. In CS6, unexpected behavior was observed in O_1 and O_2 in two separate clusters, namely cluster0 and cluster1, resulting in a CTS rating of *Low*. Similarly, in CS7, unexpected behavior was observed in O_1, O_2, and O_3 in two separate clusters, resulting in a CTS rating of *Low*. Finally, for CS8, the latency of the service did not behave as expected in all three clusters, leading to a CTS rating

of *Medium*. Based on these observations, we can conclude that the microservice has an issue with delivering good latency.

TABLE VIII
MICROSERVICE TRUST SCORE AND SERVICE PROVIDER TRUST SCORE
(L: LOW, M: MEDIUM, H: HIGH)

	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8	CS9
CTS	H	M	L	L	L	L	L	M	M
SPTS	H	M	L	L	L	L	L	M	M

c) Trend Variations of Instance Trust Score and SLA Violation Rate: For both maps, the parameters are the same as for the ITS-TV map of use case n°1. We evaluate the maps with the same metrics. Table IX summarizes the results for the two SOM maps.

TABLE IX
SOM: EVALUATION

Metrics	ITS-TV map	SVR-TV map
Quantization error	0.013	0.23
Topographic error	0.02	0.026
Silhouette score	0.65	0.78
Distortion	0.75	0.65
Neighborhood preservation	0.92	0.75

As shown, the quantization error and topographic error are both relatively low (0.013 and 0.02 for the first map and 0.23 and 0.026 for the second map), indicating that the SOM preserves the topological and spatial relationships between the input data points. The silhouette score is relatively high (0.65 for the first map and 0.78 for the second map), indicating that the clustering obtained by the SOM is of good quality. Distortion is 0.75 for the first map and 0.65 for the second map, which suggests that the data points within each cluster are tightly packed around their cluster center. Finally, the neighborhood preservation is 0.92 and 0.75, which indicates a high level of preservation.

Interpretation of the ITS-TV Map. The ITS-TV map is presented in Figure 9. We defined five forbidden areas and four warning areas. Each area is accompanied by a color code and a description of the anomaly. In the first forbidden area, in brown, the SLA'_0 exceeds the commitment by 4%. In the second forbidden area, in purple, the SLA'_2 exceeds the commitment by 21%. In the third forbidden area, in red, the SLA'_1 with `core-command` is 32% higher than the committed value. In the fourth forbidden area, in pink, the SLA'_1 with `UI` is 49% higher than the committed value. In the fifth forbidden area, in cyan, SLA'_1 with `device-mqtt` is 52% higher than the committed value. In the first warning area, in orange, the SLA'_1 with `core-command` is about 14% higher than the committed value. In the second warning area, in yellow, the SLA'_1 with `UI` is 15% higher than the committed value. In the third warning area, in blue, the SLA'_0 is 1% above the committed value. In the fourth warning area, in gray, the SLA'_2 is 8% higher than the committed value.

Operational Phase for the ITS-TV Map. To show how the map can be employed in practical situations, we establish a scenario. Initially, the service is functioning normally as expected. However, over time, the error rate gradually deteriorates from 0 to 0.53 and eventually to 0.75, while the latency improves from 98.98 ms to 96.23 ms and then to 95.14 ms. The input data are then directed toward the warning area n°4, an alert signaling the issue is triggered. Later on, the service returns to normal but with a slight increase in latency. However, it is suddenly disrupted due to an increase in latency with the `core-command` microservice, the input data enters the forbidden area n°3, and an alert is triggered. Finally, the service returns to normal and behaves as expected. The path created by the various inputs is visualized in Figure 9.

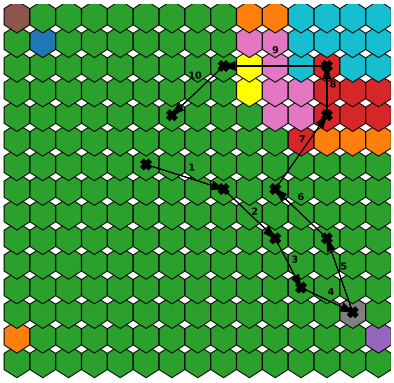


Fig. 9. ITS-TV Map with data insight.

Interpretation of the SVR-TV Map. As illustrated in Figure 10, we defined five forbidden areas and three warning areas. This first forbidden area, in blue corresponds to a probability of over 60% that SLA'_1 with `core-command` will be violated. In the second forbidden area, in brown, the probability that SLA'_0 will be violated is over 55%. In the third forbidden area, in red, there is a probability of over 60% that SLA'_2 will be violated. In the fourth forbidden area, in black, there is a probability of over 62% that SLA'_1 with `UI` will be violated. In the fifth forbidden area, in purple, there is a probability of over 62% that SLA'_1 with `device-mqtt` will be violated. In the first warning area, in orange, there is a probability of about 22% that SLA'_0 will be violated. In the second warning area, in cyan, there is a probability of about 14% that SLA'_1 will be violated. In the third warning area, in yellow, the probability of violating SLA'_2 is about 16%.

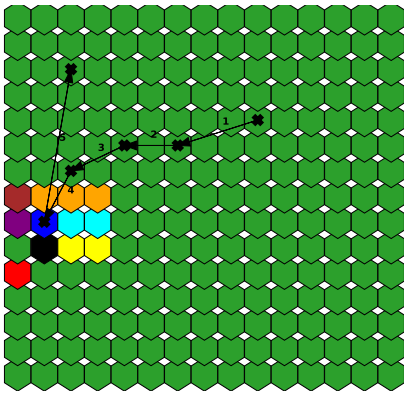


Fig. 10. SVR-TV Map with data insight.

Operational Phase for the SVR-TV Map. To demonstrate how the map can be applied in practical situations, we establish the following scenario: the service performs according to

expectations initially. The input data is projected in the green area. Then, there is a slight deterioration in latency. The projected data gradually shifts toward the forbidden area n°1. After the fifth projection, an alert is triggered as the input data is projected into the forbidden area n°1. At the end, the neuron is projected to the green area as the service resumes normal operation. The trajectory created by the input data is visualized Figure 10.

d) Financial Exposure to Performance Risk:

Highlight. The FEPR makes sense for the composition of microservices, not for an isolated microservice. For that, we first perform a SLA composition process with all the microservices using decision rules such as `maximum` and `minimum`. We reused the scenario defined for the operation phase of the SVR-TV map. The penalties for the availability, the latency and the error rate are already defined in Table V.

Result. Initially, the FEPR remains at 0$ as the service operates as expected. However, over time, there is a slight degradation in latency, which raises the risk of SLA violations and causes the FEPR to decline to -100$ and eventually to -1000$. The error rate also starts to deteriorate, further increasing the risk of SLA violations and resulting in an elevation of the FEPR to -1100$ and -2000$. Eventually, the service returns to normal operation, which reduces the risk of SLA violations and leads to a decrease in the FEPR to -900$ and back to 0$.

V. Discussion

We finalize our assessment by discussing some practical remarks in this section. Beginning with the ITS metric, the MLP demonstrated favorable outcomes for both use cases and accurately identified the trust level in various scenarios. Nevertheless, a significant drawback of utilizing this method is the necessity to train a model for each service class. Furthermore, since we operate in a dynamic environment, it is crucial to retrain the model to consider variations in the fundamental data distribution, adapt to novel patterns and trends, and enhance the model's overall effectiveness and precision. These two adverse points can be computationally expensive and time-consuming.

During the evaluation, both MLP models had a size of approximately 16MB. Using an Intel® Xeon® W-2133 Processor with 32GB RAM, the first use case model required 42 seconds to train, while the second model took slightly longer, about 55 seconds. This training time is considered reasonable.

For CTS and STS, we tried to use case studies to scan the most convincing cases. However, we only had three clusters at our disposal for the Edgex use case, so we did not have enough data to show the usefulness of k-means algorithm. Also, it can be interesting to explore alternative clustering algorithms that are better suited, for example, k-mode [18] or hierarchical clustering [19]. Additionally, it may be helpful to validate the clustering results using other metrics and visualizations, such as silhouette plots, to ensure that the clustering is meaningful and useful for the specific use case.

One of the main limitations of using SOM in trending the variation of the ITS and the SLA Violation Risk is their inability to adapt to changes in the underlying data distribution over time. This means that if the input data changes significantly or new data is introduced, the original SOM may no longer accurately represent the data and its performance may degrade. Another limitation of SOM in dynamic environments is their sensitivity to initial conditions and the specific parameters used during training. This implies that the resulting SOM may not always converge to the optimal solution. To overcome these limitations, researchers have proposed various modifications to the SOM algorithm, such as incorporating adaptive learning rates, incorporating online learning techniques, and using incremental training approaches [20]. These modifications can be integrated into a new version of the LAS as future work.

For the FEPR, we manage to demonstrate the correlation between the SLA Violation Risk and the Financial Risk Exposure with the two use cases. However, it is important to note that this metric should be presented in a clear and understandable manner, and any potential biases or limitations of the metric should be thoroughly addressed and discussed. Additionally, the metric should be used as a supplement to, rather than a replacement for, human judgment and expertise in the legal field.

VI. Related work

Liability analysis in a multi-actor and dynamic microservices architecture can be challenging due to the system complexity and the involvement of multiple actors. The research on liability in microservice architecture is still a developing field, and more studies are needed to fully understand and address the legal and technical challenges. Similarly, few papers address the research question of accountability in cloud computing.

The accountability literature extensively covers the use of data management tools to ensure data protection, privacy, security, and regulatory compliance. For example, Thiago Rodrigues *et al* [21] proposed the Cloudacc framework to ensure accountability and trust in federated cloud environments. It combines cloud and blockchain technologies to create a distributed and transparent mechanism for cloud providers to record and share information about their services and operations. Also, the A4Cloud project [22] proposes tools and models that provide users with greater control, transparency, and enforcement capabilities over the use and protection of their data in the cloud. Moreover, the authors of [23] propose a data-centric logging approach to improve accountability and security in cloud computing. Their four-stage framework includes standardizing data transaction definitions, real-time analysis for detecting security threats, and generating reports to help customers understand their data transactions. Finally, the European Telecommunications Standards Institute (ETSI) defines in the GS NFV REL 005 document principles for accountability management and presents a Quality Accountability Framework for ensuring the quality of NFV implementations and establishing accountability mechanisms. To the best of our knowledge, no previous paper has addressed the importance of providing indicators to handle liability in the cloud architecture. In addition, previous papers focus mainly on the accountability and transparency of cloud service providers in managing and protecting user data.

Since our framework proposes indicators based on the SLA concept, we have also reviewed the scientific literature on this topic. SLA verification based trust is a widely used method for building trust in cloud computing environments. It employs SLAs to establish and measure trust between providers and consumers. Valero *et al.* present in [24] a trust framework for reliable stakeholder selection in a 5G marketplace. It includes a reputation-based model with four modules: Information gathering, Trust computation, Trust storage, and Continuous update. The Continuous update module introduces an SLA-driven reward and penalty system to adjust trust scores based on breach predictions, detections, and violations. In [25], the authors propose a trust model empowering Cloud Service Providers (CSPs) to assess trust for participation in reliable Cloud federations. It relies on feedback and CSPs' SLAs, extracting Quality of Protection (QoP) attributes from SLA documents to gauge security and privacy levels. The model computes an aggregated trust value using this information. Moreover, Li *et al.* propose in [26] a Cloud-Trust model that utilizes trust parameters like security, availability, and reliability to evaluate cloud services' adherence to SLAs objectively. The entire model revolves around the compliance and monitoring of SLA signed with the end user or with the customer. Also, Chang et al. [27] propose a multi-dimensional trust model for fog computing that considers application, peer, and auditor perspectives. Parameters like availability, response time, throughput, and security are used to assess Fog service provider trustworthiness, with adjustable weights for each perspective based on application requirements. Finally, J.Bendriss *et al.* present in [28] a framework for cognitive SLA enforcement of networking services involving Virtual Network Functions (VNFs) and using ANN. This framework is designed to efficiently manage and anticipate SLA breaches. The framework identifies correlations in historical data and predicts future resource usage, which helps optimize resource utilization and reduces the risk of SLA violations. No implementation or evaluation is provided for the models proposed in these papers. Moreover, the models primarily serve to compare cloud providers based on the services provided and the level of trust they instill. In contrast, the frameworks in question are more specifically tailored towards cloud computing and 5G and lack the generality present in the framework that we propose. Our framework provides trends of SLA Violation Rate to predict SLA violation based on field observation, which is not provided by other frameworks, in addition to liability and trust indicators.

VII. Conclusion and Future Work

With microservices, multiple actors can contribute to a service, making it challenging to manage the liability and trust associated with each component. A liability framework can help in addressing this challenge by providing a comprehensive assessment of the liability and trust of the various actors involved in the microservices system.

This paper introduces the LASM Analysis Service (LAS), a framework for analyzing liability and trust in multi-actor dynamic microservices. Utilizing ML, the LAS framework calculates three types of liability and trust metrics: Commitment Trust Scores, which assess the trust that an instance, all instances of a microservice, or all microservices of a provider will perform as expected based on SLA commitments; Financial Exposure, which measures the potential monetary loss for the overall microservice architecture provider with the current composition of microservices; and Commitment Trends, which monitors trends of SLA Violation Rates and Instance Commitment Trust to predict violations. The framework has been implemented on a Kubernetes platform. We apply our framework to evaluate two services in different scenarios and case studies, namely a network service that simulates the behaviour of the PacketFabric service [15] and Edgex service for IoT edge computing. The results demonstrate the effectiveness of the LAS in accurately computing the trust and liability metrics for both use cases. Finally, we discuss our results by considering its impact on computation time and its limitations, as well as potential enhancements to the underlying algorithms so we can better address the effectiveness of the solution.

As part of our future work, one research direction is microservice dependencies, which could influence responsibility-related indicators. We plan to investigate this aspect further. We also aim to explore additional metrics from the SOM map and test the LAS in various use cases beyond our current ones, like a 5G service involving IoT and Edge computing services.

References

[1] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2016, pp. 44–51.

[2] C. Gaber, J. S. Vilchez, G. Gür, M. Chopin, N. Perrot, J.-L. Grimault, and J.-P. Wary, "Liability-aware security management for 5G," in *2020 IEEE 3rd 5G World Forum (5GWF)*, 2020, pp. 133–138.

[3] Y. Anser, C. Gaber, J.-P. Wary, S. N. M. García, and S. Bouzefrane, "TRAILS: Extending TOSCA NFV profiles for liability management in the cloud-to-IoT continuum," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 321–329.

[4] O. Kalinagac, W. Soussi, and G. Gür, "Graph based liability analysis for the microservice architecture," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022.

[5] E. M. Chrystel Gaber, Vinh Hoa La, "Liability management in a 5G environment," in *INSPIRE-D4.4*, 2022.

[6] M.-C. Popescu, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, Jul. 2009.

[7] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, pp. 1–6, 1990.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, jun 2002.

[9] T. Elhassan, A. M, A.-M. F, and M. Shoukri, "Classification of imbalance data using tomek link combined with random under-sampling as a data reduction method," *Global Journal of Technology and Optimization*, vol. 01, 01 2016.

[10] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," Dec. 2019.

[11] D. Marutho, S. Hendra Handaka, E. Wijaya, and Muljono, "The determination of cluster number at k-mean using elbow method and purity evaluation on headline news," in *2018 International Seminar on Application for Technology of Information and Communication*, 2018, pp. 533–538.

[12] K. Potdar, T. Pardawala, and C. Pai, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *International Journal of Computer Applications*, vol. 175, pp. 7–9, Oct. 2017.

[13] G. Vettigli, "Minisom: minimalistic and numpy-based implementation of the self organizing map," 2018. [Online]. Available: https://github.com/JustGlowing/minisom/

[14] F. Forest, M. Lebbah, H. Azzag, and J. Lacaille, "A Survey and Implementation of Performance Metrics for Self-Organized Maps," *arXiv e-prints*, p. arXiv:2011.05847, Nov. 2020.

[15] (2022) PacketFabric's terms and conditions. [Online]. Available: https://packetfabric.com/terms-and-conditions

[16] O. Kalinagac, W. Soussi, Y. Anser, C. Gaber, and G. Gür, "Root cause and liability analysis in the microservices architecture for edge iot services," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 3277–3283.

[17] S. v. Buuren and K. Groothuis-Oudshoorn, "Mice: Multivariate Imputation by Chained Equations in R," *Journal of Statistical Software*, vol. 45, pp. 1–67, Dec. 2011.

[18] M. Á. Carreira-Perpiñán and W. Wang, "The K-modes algorithm for clustering," *arXiv e-prints*, p. arXiv:1304.6478, Apr. 2013.

[19] V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu, "Hierarchical clustering: Objective functions and algorithms," *CoRR*, vol. abs/1704.02147, 2017.

[20] N. Rougier and Y. Boniface, "Dynamic self-organising map," *Neurocomputing*, vol. 74, no. 11, pp. 1840–1847, May 2011.

[21] T. Rodrigues, P. Endo, D. Beserra, D. Sadok, and J. Kelner, *Accountability for Federated Clouds*, 01 2018, pp. 569–583.

[22] C. Fernandez-Gago, V. Tountopoulos, S. Fischer-Hübner, R. Alnemr, D. Nuñez, J. Angulo, T. Pulls, and T. Koulouris, "Tools for cloud accountability: A4cloud tutorial," in *Privacy and Identity Management for the Future Internet in the Age of Globalisation*, J. Camenisch, S. Fischer-Hübner, and M. Hansen, Eds. Cham: Springer International Publishing, 2015, pp. 219–236.

[23] R. Ko, M. Kirchberg, and B. Lee, "From system-centric to data-centric logging-accountability, trust & security in cloud computing," *2011 Defense Science Research Conference and Expo (DSR)*, 08 2011.

[24] J. M. J. Valero, V. Theodorou, M. G. Pérez, and G. M. Pérez, "Sla-driven trust and reputation management framework for 5g distributed service marketplaces," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–13, 2023.

[25] A. Kanwal, R. Masood, and M. A. Shibli, "Evaluation and establishment of trust in cloud federation," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '14. Association for Computing Machinery, 2014.

[26] X. Li and J. Du, "Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing," *IET Information Security*, vol. 7, no. 1, pp. 39–50, 2013.

[27] V. Chang, J. Sidhu, S. Singh, and R. Sandhu, "SLA-based multi-dimensional trust model for fog computing environments," *Journal of Grid Computing*, vol. 21, no. 1, p. 4, 2022.

[28] J. Bendriss, I. G. Ben Yahia, P. Chemouil, and D. Zeghlache, "Ai for sla management in programmable networks," in *DRCN 2017 - Design of Reliable Communication Networks; 13th International Conference*, 2017, pp. 1–8.

Yacine Anser received the M.Sc. degree in Computer Engineering in Network and Cybersecurity from INSA Toulouse (Institut National des Sciences Appliquées) in 2020. Obtained a Ph.D. from CNAM (Conservatoire National des Arts et Métiers) Paris, France, at the CEDRIC laboratory in 2024, in collaboration with Orange Lab Caen. Currently engaged in postdoctoral research at KTH (Kungliga Tekniska Högskolan) in Stockholm, Sweden. Research interests lie at the intersection of cybersecurity, machine learning, and network systems.

Chrystel Gaber received her PhD from University of Caen in 2013 in Computing Systems. After an experience as project coordinator and R&D engineer in Fime, she joined Orange as a researcher & project coordinator. She contributes to several projects related to cyber-physical security, IoT device management and certification. She participated in the FP7 project MASSIF and ensured the coordination lead of the CELTIC-PLUS project ODSI. She represented Orange in GSMA work groups related to the certification of integrated SIMs and the accreditation of SIM production sites. Currently, she contributes to the H2020 European project INSPIRE-5GPlus and leads the franco-german project TinyPART.

Samia Bouzefrane received the Ph.D. degree in computer science from the University of Poitiers, France, in 1998. After four years at the University of Le Havre, France, she joined the CEDRIC Lab of Conservatoire National des Arts et Métiers (Cnam), Paris, in 2002. She is currently full professor in Cnam and the head of CEDRIC Lab. She is the coauthor of many books (Operating Systems, Smart Cards, and Identity Management Systems). She has coauthored more than 120 research articles. Her current research interests include the Internet of Things and security using AI techniques.

Jean-Philippe Wary has been a Research Program Director with Orange Laboratories, since 2011, in charge of infrastructures security research for 5G and the IoT topics. He was with SFR (French Mobile Operator) as a Security Expert, for 15 years and the Chief Information Security Officer of networks and services. He was also with Alcatel (real time, telecom, security, and electronic war), for eight years.

Méziane Yacoub is an associate professor at the CNAM in Paris. He received his PhD in computer Science in 1999 from University of Paris 13 (France). His current research interests include self-organizing maps, graph neural networks, anomaly detection and explanation.

Onur Kalinagac Onur Kalinagac received his B.S. degree in Electrical and Electronics Engineering and his M.S. degree in Computer Engineering from Bogazici University, Istanbul, Türkiye, in 2015 and 2022, respectively. He served as a research assistant at the Institute of Computer Science, Zurich University of Applied Sciences, Winterthur, Switzerland, from March 2022 to January 2023. His research interests include UAV networks, task offloading, and root cause analysis in microservices.

Gürkan Gür is a senior lecturer at Zurich University of Applied Sciences (ZHAW) – Institute of Computer Science (InIT) in Winterthur, Switzerland. He received his B.S. degree in electrical and electronics engineering in 2001 and Ph.D. degree in computer engineering in 2013 from Bogazici University in Istanbul, Turkey. His research interests include Future Internet, information security, 5G and Beyond networks, and critical infrastructure protection. He is a senior member of IEEE and a member of ACM.