IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xxx

Interactivity anomaly detection in remote work scenarios using LSTM

JESUS ARELLANO-USON[1], EDUARDO MAGAÑA[1,2], DANIEL MORATÓ [1,2] AND MIKEL IZAL[1,2]

[1]Department of Electrical, Electronic and Communications Engineering, Public University of Navarre at Arrosadia Campus, 31006 Pamplona, Spain
[2]Institute of Smart Cities, 31006 Pamplona, Spain

Corresponding author: Jesus Arellano-Uson (e-mail: jesus.arellano@unavarra.es)

This work was supported by Spanish State Research Agency project number PID2019-104451RB-C22/AEI/10.13039/501100011033

ABSTRACT In recent years, there has been a notable surge in the utilization of remote desktop services, largely driven by the emergence of new remote work models introduced during the pandemic. These services cater to interactive cloud-based applications (CIAs), whose core functionality operates in the cloud, demanding strict end-user interactivity requirements. This boom has led to a significant increase in their deployment, accompanied by a corresponding increase in associated maintenance costs. Service administrators aim to guarantee a satisfactory Quality of Experience (QoE) by monitoring metrics like interactivity time, particularly in cloud environments where variables such as network performance and shared resources come into play. This paper analyses anomaly detection state of the art and proposes a novel system for detecting interactivity time anomalies in cloud-based remote desktop environments. We employ an automatic model based on LSTM neural networks that achieves an accuracy of up to 99.97%.

INDEX TERMS remote work, interactivity time, anomaly detection, LSTM, cloud-based interactive applications, remote desktop, QoE.

I. INTRODUCTION

CLOUD-BASED interactive applications (CIA) are those applications whose major part of load runs in the cloud with strict end-user interactivity requirements. They are also known as cloud-based distributed interactive applications (CDIAs) [1] or real-time interactive applications (RIAs) [2]. Examples of CIA are remote desktop services, cloud gaming services and interactive web applications [1]. In remote desktop services, user interaction occurs by pressing a key or clicking a mouse. The response corresponds to an update of the screen content so fast that the user cannot perceive that the application is not running locally. Services such as video-on-demand or voice-over IP are not considered CIA because they do not have interactions as defined above. These services utilize audio/video data streams with temporal requirements but without specific user interaction requirements with the service.

In the CIA paradigm, users can connect from a local device to a remote one as if they were sitting in front of the remote device. The local device may have limited computational resources and is referred to as a thin client in the context of remote desktop scenarios. In the context of interactive web applications, the local device can be a web browser running on any device, while in cloud gaming scenarios, it can be a gaming console, computer, smartphone, tablet, etc. The remote device is typically a virtual machine running on a shared server, although it can also be a dedicated computer. The thin client captures mouse movements, clicks, and keyboard keystrokes to send them to the remote device, which is responsible for executing the applications. This paradigm facilitates user mobility, allowing access from any device and location, with minimal reliance on an internet connection and lowered requirements for management and maintenance, leading to cost savings.

The COVID-19 pandemic imposed remote work measures in favour of public health. Studies [3] [4] suggest that approximately 40% of large and small companies expected 40% or more of their workers to adopt remote work after the health crisis. Ultimately, of those workers before the COVID-19 pandemic, nearly 50% became remote workers [5]. It is estimated that by 2025, 70% of employees will work remotely at least five days a month [4]. Many of the services that enable remote work are within the scope of interactive cloud-based applications. Specifically, the use of

remote desktops dramatically increased due to this situation. In the information technology sector, the increase in remote desktop deployment was 258% in 2020 [6], and spending on remote desktop solutions doubled in 2021 [7]. Meanwhile, other types of CIA such as cloud gaming solutions are also booming, with this industry expected to reach a value of $3.107 billion by 2024, representing a 54% increase from 2019 [8].

These data reveal that CIAs such as remote desktops are increasingly prevalent and require the development of solutions to monitor Quality of Experience (QoE) [9] [10]. It's crucial for cloud service providers to grasp users' perceptions of the services they offer [11] [12], especially as both personal and business applications transition to the cloud, making perceived service quality a key differentiator among providers [13]. Among the metrics that characterize QoE, the interactivity time, the number of frames per second, and the image resolution stand out. Interactivity time [14] is the time elapsed between a user interaction via keyboard or mouse and the on-screen update with the result of the user interaction. This metric is also referred to as responsiveness [15]. CIAs have particular interactivity requirements and therefore the interactivity time metric is, to a larger extent, responsible for the user experience. While these requirements are present in applications running locally, they become critical in cloud-based interactive services because there is a network between users and the cloud. The network introduces additional elements like latency, jitter, packet loss, and bandwidth constraints. Moreover, server computational resources are frequently shared among multiple users, potentially impacting the end-user experience.

The literature suggests that further work is needed on remote work communications [16]. In this work, we build upon the extraction of interactivity time from a previous study [14], using the system in a national parcel delivery company with a network of remote desktops across different offices. We collected data from real users to design a scheme for the automatic detection of anomalies in interactivity time. While there have been previous attempts to measure QoE in remote desktops, as we will elaborate in state of the art section, they frequently exhibit significant limitations.

This study presents a pioneering system aimed at preemptively identifying operational issues in remote desktops before they become perceptible to the user. Our approach involves the implementation of an automated model employing Long Short-Term Memory (LSTM) neural networks and supervised learning. Notably, no previous work has addressed anomaly detection during interactivity times, marking a significant gap in the existing research landscape. We introduce, for the first time, available options for anomaly detection and propose a viable solution based on a comprehensive literature analysis, leveraging LSTM neural networks.

The main contributions of this paper are as follows:

- We conduct an extensive analysis of the state of the art in anomaly detection. We propose a novel and clear categorization that can assist researchers in deciding which method is more suitable for anomaly detection in this or any other domain.
- We evaluate and identify the most suitable solutions from the literature for quantifying QoE in CIAs. Additionally, we highlight the drawbacks of current solutions.
- We propose an innovative system for detecting anomalies in interactivity time in cloud-based remote desktop systems. This system enables service administrators to assess the real-time QoE of their deployments.
- For the implementation of the anomaly detection system, we employ LSTM neural networks due to their inherent ability to capture long-term dependencies in time series data.
- We present and discuss new research directions derived from this paper with the intention of facilitating further investigations.

The rest of the document is structured as follows: in Section II, we evaluate the state of the art in QoE quantification and anomaly detection. In Section III, we explain the test environment we use and present the data used for training and evaluating the anomaly detection model. In Section IV, we describe the proposed LSTM model, the features it employs, and the parameter adjustments we made. In Section V, we evaluate the proposed model, achieving an accuracy of up to 0.9997, a precision of up to 0.9367, a recall of up to 0.9779, and an F1 score of up to 0.9569. Finally, Section VI concludes the paper.

II. STATE OF THE ART

In this section, we delve into the analysis of the state of the art. Firstly, we explore the current state of the art related to acquiring interactivity time metrics. To achieve this, first subsection is dedicated to outlining the primary options identified in the literature. We elaborate on their operation, advantages, and disadvantages. Also, we focus on the state of the art in anomaly detection. For this purpose, the second subsection conducts an extensive categorisation of existing anomaly detection methods in the literature. We anticipate that this categorisation will serve as a valuable starting point for future implementations of anomaly detection, regardless of the application domain. Finally, last subsection summarizes the key conclusions from the two preceding subsections, justifying the procedures followed and the proposed anomaly detection model in this paper.

A. TECHNIQUES FOR MEASURING INTERACTIVITY TIME

The literature currently presents two primary methodologies for assessing QoE of remote desktops through the quantification of interactivity time: Slow-motion benchmarking [17] [18] and thin client latency analysis (TeCLA) [14]. Both approaches aim to measure the interactivity time of remote desktops. For more detailed and comprehensive information, we recommend consulting the original papers.

TABLE 1: Comparison of techniques for measuring interactivity time

	Proposed Methodology	Methodology Drawbacks	Methodology Advantages
Slow-Motion Benchmarking [17] [18]	• Observing peaks in network traffic to infer interactions from the lightweight client and server response.	• Provides an approximate measure of interactivity time • Requires the instrumentation of applications to prevent more than one unresponsive interaction • Does not allow obtaining real-time measurements • Does not take into account the processing delay introduced by the thin client	• Widely adopted and well-established • Easy to implement • Independent of the protocol or application used
Thin Client Latency Analysis (TeCLA) [14]	• Observing changes on the screen, as well as keyboard and mouse interactions, to identify when screen updates are received by the thin client.	• Requires installing an agent on the thin client	• Provides real-time interactivity metrics • Enables real-time metric acquisition • Takes into consideration all elements involved in the infrastructure • Independent of the protocol or application used

Slow-motion benchmarking provides only an estimation of interactivity time based on the patterns of network packets exchanged between the thin client and the cloud server. When a user interacts with an application through a remote desktop, the thin client sends a request to the cloud server. This request generates an increase in network traffic. When the application located on the cloud server produces a response, it sends it back to the thin client, generating another spike in network traffic. Slow-motion benchmarking monitors the network traffic to obtain the time elapsed between the two traffic peaks. If slow-motion benchmarking captures the traffic near the thin client, this elapsed time is an approximation of the time the user perceives from interacting via keyboard or mouse with the thin client to perceiving a response on the screen. However, the methodology requires instrumenting the applications on the thin client to prevent a new user interaction until a response to the previous request is received. Therefore, it is not suitable for measuring a user's real-time QoE, but only serves to characterize the scenario under controlled conditions. Several works in the literature employ the methodology described by slow-motion benchmarking [19] [20] [21].

TeCLA provides another method to quantify QoE in remote desktops, regardless of the protocol used, and without the need to instrument the applications. The system is based on an agent running on the thin client that collects timestamps of keyboard and mouse interactions, as well as timestamps of screen changes. By correlating these timestamps, TeCLA extracts the interactivity time for each user interaction. The interactivity time is, therefore, the sum of the time taken to capture the interaction on the thin client, transmit that interaction to the server over the network, process it by the remote desktop system on the server, the time it takes for the application to generate the response, the generation of the screen refresh and its compression, the transmission of that refresh over the network, and the display of the refresh on the screen.

In summary, Table 1 shows the two primary methodologies from the literature, along with their respective advantages and disadvantages.

While these two proposals from the literature are considered the most effective for capturing interactivity time, the dynamic nature of CIA service operations necessitates more than static thresholds to gauge sufficient QoE. Administrators require advanced anomaly detection and trend identification systems to effectively manage these dynamic environments. In the next subsection, we explore the existing options in the literature to address these needs.

B. TECHNIQUES FOR ANOMALY DETECTION

Anomaly detection involves identifying individual samples or sets of samples that are substantially different from what would be expected [22] [23]. Due to the novelty of the interactivity time metric, to the best of our knowledge, there are no previous works in the literature that specifically address anomaly detection in time series of this nature. However, the development of algorithms for anomaly detection is a well-established research field with over 30 years of work [23] in various domains such as network traffic monitoring and computer systems, cybersecurity banking sector, and

TABLE 2: Comparison of anomaly detection methodologies.

Method	Advantages	Drawbacks	Operation	Models
Heuristic Models	• Easy implementation	• Requires a static threshold • Does not adapt well to complex temporal patterns	• Method selects maximum and minimum thresholds and compares them with samples	• Static Thresholds • Statistical Tests • Baseline Models • Density-Based Models
Statistical Models	• Good performance with data that follows a specific statistical model	• Does not take into account temporality and periodic patterns	• Method assumes a statistical model and compares predictions with actual observations	• SARIMAX Models • Exponential smoothing
Machine Learning	• Efficient in multidimensional environments	• Requires labeled data for training	• Method trains models on labeled datasets, allowing them to recognize patterns and relationships. Once trained, these models can make predictions or classifications on new, unseen data.	• Clustering • Nearest Neighbour • Classification • Ensemble
Deep Learning	• Assumes no specific distribution	• May require large amounts of data to train and computational power	• Method employs neural networks with multiple layers (deep neural networks) to learn intricate patterns and representations.	• Convolutional Neural Networks (CNNs) • Generative Adversarial Networks (GANs) • Recurrent Neural Networks (RNNs)

healthcare [24].

By definition, anomalies are rare occurrences because if they were common, they would change the typical pattern of the data. One immediate option for handling observations or sets of observations that deviate from the common pattern or distribution is to model the normal observations. These anomalies can be global or local in nature, leading the literature to distinguish between three types of anomalies [24] [22] [23] [25].

- Point anomaly: An observation whose value is atypical at any point in time.
- Contextual anomaly: An observation or set of observations that deviate from the usual pattern not because of their value but due to the context (date, time, and/or adjacent observations).
- Collective anomaly: Groups of observations where individual observations alone are not anomalous, as they can occur in a usual manner separately and in their temporal context, but not consecutively.

There are several works in the literature that address the classification of anomaly detection methods [23] [26] [27] [25] [28] [24] [22]. From these studies, we extract some common methods:

1) Heuristic Models:

In this type of method, a maximum and/or minimum threshold is selected, and it is compared when one or several samples of the time series fall outside the limits defined by these thresholds. Within these models, we find:

- Static Thresholds: The analyst manually selects the threshold for each time series. They require prior knowledge of the monitored metric. If the usual values of the metric change, the analyst must repeat the process. If the metric under study varies significantly in magnitude following a periodic pattern, it becomes necessary to define multiple thresholds based on date/time. Otherwise, the ability to detect contextual anomalies may be limited.
- Statistical Tests: The threshold is calculated based on the data series model. For example, assuming the data follows a Gaussian distribution and setting the thresholds at the mean ± 3 standard deviations or considering percentiles [26] [29] [30]. This approach does not take into account the temporal component and the potential periodic nature of the samples.
- Baseline Models: For data with periodic temporal patterns, the historical data can be used to estimate normality over recent periods and apply thresholds or statistical tests for anomaly detection. However, if the pattern of normality varies over time, the baseline may not be representative. Additionally, if past anomalies represent a significant deviation in the magnitude of the monitored metric, this can affect statistical descriptors such as mean and standard deviation, making it challenging for the baseline to function correctly [31].

- Density-Based Models: These models involve constructing a model that represents the density of observations or the probability of encountering certain values. Less probable values are identified as anomalies [23] [26] [25].

2) Statistical Models:

These models assume a specific model or distribution of the data they try to fit. They involve fitting a statistical model that allows for evaluating whether the underlying process generating the data series has undergone any changes. Often, these models use statistical fitting to predict future samples and compare their prediction with the actual observation [24]. The error made in this process can determine whether the sample is an anomaly. Within these models, we find:

- SARIMAX Models: These models are widely used in the field of statistics for sample prediction. They take into account previous samples (autoregressive component), errors made in predicting previous samples (moving average component), and the possible variation in magnitude of the data series (differencing transformation component) [24].
- Exponential smoothing: It encompasses methods that provide an estimation of the current observation based on a weighted average of previous observations, following an exponentially decreasing distribution according to the age of the samples. This allows the model to adapt more easily to new patterns in the time series. However, in order to work properly with periodic time series, manual adjustments to the model are required [29].

3) Machine learning:

These algorithms have been developed for use with multidimensional observations without temporal reference. However, sliding windows can be used with a certain number of observations or the time frame to which the sample belongs can be included in the observation in the case of patterns with periodic behaviour. Within these models, we find:

- Clustering: this technique assigns each observation to different groups or clusters based on their relative positions in the sample space. The distances of the samples to the centre of their cluster (centroid) or the size and density of the clusters provide a metric that can be used to assess the abnormality of a sample. However, the anomalous and non-anomalous samples need to be highly distinguishable, making it difficult to identify collective or contextual anomalies using these techniques [23] [25]. Some methods within this category include K-Means or DBSCAN.
- Nearest Neighbour: These techniques are based on the assumption that normal observations, compared to anomalies, exhibit closer proximity to other observations or are situated in regions of higher density.
- Classification: These methods involve training a model to differentiate between normal and anomalous samples. However, it is necessary to have a sufficient amount of labelled data for the algorithm to undergo training. Examples of algorithms belonging to this category include One-Class Support Vector Machine (OC-SVM), K-means, hierarchical clustering, or Decision trees.
- Ensemble: These methods use a combination of simpler algorithms to achieve better performance than each algorithm would achieve individually. The ensemble can consist of multiple instances of the same algorithm or model, or it can be formed by different algorithms working together [22]. Some well-known ensemble models include *Isolation Forest*, *Robust Random Cut Forest*, *Adaptive Boosting*, *Gradient Boosting*, and *Extreme Gradient Boosting*.

4) Deep Learning:

These algorithms utilize neural networks for anomaly detection. Similar to machine learning algorithms, this type of approach does not require assuming any specific distribution or process responsible for generating the observations [24]. Within the realm of deep learning models, we find:

- Convolutional Neural Networks (CNNs): These neural networks are commonly used in image processing and object detection tasks. They operate by applying mathematical convolutions on n-dimensional matrices using a sliding kernel matrix.
- Autoencoders: This type of neural network produces a simplified yet representative version of the input at the output, enabling the reconstruction of the original values. It is commonly employed by training the network on non-anomalous samples and assuming that anomalous values will result in higher reconstruction errors, facilitating their detection.
- Generative Adversarial Networks (GANs): GANs involve the training of two independent neural networks, the generator and the discriminator, in a competitive manner. The networks aim to optimize their respective metrics by improving their outputs through iterative training. In the context of anomaly detection in time series data, the generator network is trained to produce samples that resemble the input time series. The discriminator network is then trained to differentiate between real samples and artificially generated samples produced by the generator. Once both networks are trained, the discriminator can be directly utilized for anomaly detection (considering anomalies as artificial observations), or the difference between the generated and real samples can be used as a reconstruction error [32].
- Recurrent Neural Networks (RNNs): These neural networks employ feedback connections within their internal layers. These connections enable the networks to retain information across time, improving their performance on temporal data. Among the common recurrent layers in this type of network are LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units)

[24] [33]. One of the advantages of these recurrent architectures is their ability to not only adjust their internal parameters during training but also alter internal memory variables as they process observations during regular usage. This allows them to store information from processed observations that may be useful for subsequent ones, which is particularly relevant in the context of time series data, where observations often exhibit dependencies on immediately preceding values. These recurrent networks account for the temporality, periodicity, and repetitiveness of anomalies, making them potentially useful for detecting point anomalies, contextual anomalies, and collective anomalies.

Due to the significant increase in computational capacity in recent years, the utilization of technologies such as Machine Learning and Deep Learning for large-scale data applications has become feasible, demonstrating their successful performance in a wide range of tasks. Currently, there is a need and opportunity to develop automatic algorithms that can enhance the capabilities of existing systems [24] and allow for more efficient utilization of available human resources [34]. In this study, we aim to explore the implementation of LSTM neural networks for anomaly detection in time series of interactivity time.

For some machine learning problems, it is important to consider information from nearby time steps. In such cases, recurrent neural networks (RNNs) have shown satisfactory results [35] [36]. However, when there are long-term dependencies, as in the case of this paper [24], traditional RNN models are unable to effectively capture them [37]. This is why Long Short-Term Memory Networks (LSTM) were developed to address these issues [37]. LSTM networks are designed to overcome the limitations of traditional RNNs and are capable of modelling long-term dependencies in practice.

In summary, we have compiled the primary anomaly detection methodologies from the literature, including their respective advantages and disadvantages, in Table 2.

C. SUMMARY

In a remote desktop deployment, monitoring user QoE is crucial. Tools like Slow-motion benchmarking or TeCLA provide interactivity time metrics, but it is imperative to conduct contextual work to define the typical interactivity times for each specific deployment. However, the operation of these services is highly dynamic, and it is not sufficient to define static thresholds on interactivity time to determine if the QoE is sufficient. Administrators require more advanced systems for anomaly detection and trend identification. These systems should not only focus on discrete interactivity times, but also analyse the time series of interactivity experienced by the user. Among the two literature options, we selected TeCLA for extracting interactivity times in this study, motivated by the advantages listed in Table 1.

The nature of the input data, whether numerical data sets, server logs, user text strings, or images, substantially influences the methods employed for anomaly detection or trend identification in each respective field of study [24], [38]. There is an obvious challenge in using algorithms developed in one field for another [22]. Moreover, with the surge in simultaneous real-time monitoring [39] [40] and the unprecedented volumes of data, which now reach tens of terabytes per day in a single system [24] [34], new difficulties and limitations have emerged. In the specific case of information systems, many of the metrics used exhibit strong relationships with human behaviour patterns, creating temporal dependencies and periodicities (daily, weekly, monthly, yearly, etc.) [41], which complicates their analysis. In recent years, these systems have been experiencing constant shifts in volume and structure [42]. Additionally, the near-daily deployment of new services has significantly heightened the complexity of monitoring and alarming in such environments.

Given these challenges, the use of machine learning and deep learning solutions [43] has increased, enabling automation of parts of the analysis process. With these tools, data classification, temporal estimations, and anomaly detection can be successfully developed. For machine learning problems that necessitate information from nearby time steps, RNNs have demonstrated good results. However, in the presence of long-term dependencies, as in this case, traditional RNNs are limited. For this reason, LSTM networks were developed to overcome these limitations, enabling the effective capture of long-term dependencies in practice. Hence, based on existing literature and the specific characteristics of the data in this article (time series of interactivity time), we chose to employ LSTM neural networks.

III. DEPLOYMENT OF INTERACTIVITY TIME MEASUREMENTS

From the options discussed in the state-of-the-art analysis, we have opted to utilize the TeCLA methodology for extracting interactivity times. In this section, we provide a more detailed explanation of how we implemented this methodology in two real-world environments to gather data for developing an anomaly detection model using LSTM neural networks. It is important to note that while we have selected TeCLA for extracting interactivity times, any literature option, such as Slow-Motion Benchmarking, would be suitable as input for the LSTM model to be described in subsequent sections.

A. TECLA DEPLOYMENT

The TeCLA methodology can be deployed in two operating modes:

- Active mode: allows the emulation of user interactions from the thin client. It generates periodic keyboard or mouse inputs, enabling the collection of interactivity time metrics at regular intervals. This mode provides greater temporal visibility and does not require direct involvement from the user.
- Passive mode: This mode allows for capturing samples of the interactivity time as the user interacts with a remote desktop. The interactivity time extraction tool runs

in the background and only collects measurements when the user interacts with the thin client using the keyboard or mouse. This mode enables real-time monitoring of the actual QoE for users in the deployed scenario.

The suitability of each operating mode depends on the characteristics of each environment and the level of granularity required. In some cases, service administrators may not have the possibility to deploy in passive mode due to administrative reasons (security policies, connectivity issues) or because they are not interested in detailed QoE information for individual users in managed remote desktops. Instead, their interest lies in understanding the average perceived QoE of a generic user infrastructure. In such cases, the active mode is more suitable, as it simplifies the deployment process and provides higher temporal granularity.

Figure 1 presents the procedure to extract interactivity measures. The only difference between the passive and active modes is who triggers the interactions with the remote desktop. In the active mode, the process would be repeated periodically using an automator (e.g., every two seconds), while in the passive mode, it would be the user who determines when the next interaction occurs in their normal usage pattern.

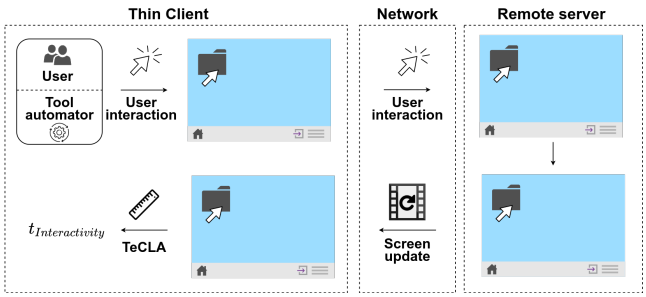

FIGURE 1: Procedure for obtaining measures of interactivity.

B. SCENARIO AND AVAILABLE DATASET

The use of both modes in different scenarios allowed us to have a representative dataset of interactivity time. We deployed the interactivity measurement tool in active mode to evaluate and monitor the remote desktop architecture in a university environment. With over a month and a half of continuous monitoring, we were able to study the behaviour of the university's Virtual Desktop Infrastructure (VDI), which includes instances based on both Linux and Windows [44]. Additionally, we deployed the measurement tool in passive mode in a national parcel company to collect QoE data from 22 real users across 11 different offices. Each user connects from their office to a remote desktop located in the central corporate data centre using Citrix [45] and Windows RDP [46]. The user's thin client has a specific installation for this type of equipment (Windows 7 Embedded Standard), and the offices are located at different distances from the data centre. Both types of deployments served as sources of information for the development of the dataset in this paper. From a network standpoint, the offices are organized in a star network configuration with connections to the central virtualization server. Each office benefits from its own independent internet access, which is shared among all users within the office. To maintain the effectiveness of our measures throughout both our experiments, we thoroughly assessed the network's available bandwidth and latency in both scenarios, eliminating any potential issues related to congestion or packet losses.

Since we opted to develop supervised models, it is necessary to label the samples through a labelling process. It is the analyst's task to determine which time intervals in the dataset should be considered as anomalies. For both data sources (active and passive interactivity time time series), we collaborated with the service administrators of the remote desktops in both scenarios to identify periods that the machine learning model should interpret as anomalies. Incorrect labelling of the training and evaluation dataset can lead to the model learning incorrect behaviours and patterns. The administrators have access to other sources of information on incidents, which helped them to identify periods in the available time series where the QoE of the remote desktop deteriorated.

Figure 2 and figure 3 represent an example of two time series labelled in passive mode and active mode respectively. The labelling was done by the service administrators, and the difference in time between samples is evident in each mode. In the active mode, the samples are approximately equispaced every two seconds, while in the passive mode, the samples are separated based on when the user interacts with the remote desktop.

A total of 22 different users were available for the passive mode dataset. We measured the available samples over a period of 20 days. We collected 109,144 keyboard or mouse employee's interactions. In the case of the active mode deployment in the university environment, we monitored 26 days, comprising a total of 1,322,969 samples. Table 3 provides an overview of both scenarios. We reduced this dataset to include only the days that had anomalies in order to have a representative dataset for training the model. As we will discuss in later sections, the features used for the proposed model take into account that the data comes from two different environments (different network, virtualization software, clients, and servers, etc.). We carefully selected the features to eliminate such variability. The measurement tool originally provided information on metrics related to the state of the thin client, such as CPU usage, RAM memory, transmitted and received bytes, and consumed bandwidth, in addition to the interactivity time metric. However, these quality of service (QoS) parameters were not the focus of our study.

TABLE 3: Summary of the samples available for training and evaluation of the models we propose.

Deployment environment	Operating mode	No. days	No. samples
University	Active	26	1.322.969
Parcel company	Pasive	20	109.144

We divide the main dataset into training and evaluation sets. In our evaluation, the dataset is randomly split by day and by user after the data fusion process, where all days of all users are concatenated. In this way, we assign each day to the training set with a probability of 60%. We use the remaining days for evaluation. We train the models with the training set, and then we evaluate their performance using the evaluation set. It should be noted that, due to the temporal nature of the dataset, this splitting of samples must be done while respecting the membership of samples to a specific day. For example, if the dataset has 10 days, 6 days will be used for training and 4 days for evaluation. All samples from the 6th day will be included in the training dataset. This splitting should be done precisely to avoid mixing samples from different days, which could lead to the network learning unrealistic temporal relationships.

From the total available days of active mode time series, we prepared the training and evaluation sets while respecting whole multiples of days as closely as possible. We allocate 60% of the samples to the training set and 40% to the evaluation set. In this case, we used a total of 272,275 samples for training and 165,504 for evaluation, resulting in a ratio of approximately 62.19% to 37.81%. Within the training set, administrators labelled a total of 759 samples as 'Anomalous,' while in the evaluation set, they labelled 499 samples.

For the passive mode dataset, once again, we prepared the training and evaluation sets while respecting integer multiples of days as closely as possible. We allocated approximately 60% of the samples to the training set and 40% to the evaluation set. In this case, we used a total of 28,992 samples for training and 17,541 for evaluation, resulting in a ratio of approximately 62.30% to 37.69%. Within the training set, administrators labelled a total of 1,040 samples as 'Anomalous,' while in the evaluation set, they labelled 474 samples.

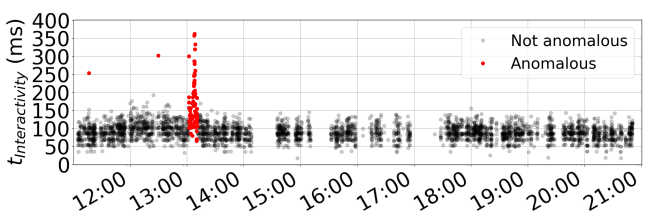

FIGURE 2: Example of labelling of passive mode interactivity time series.

The anomalies present in figures 2 and 3 seem easily distinguishable. Traditional methods such as heuristic or statistical models would have no problem in correctly categorising

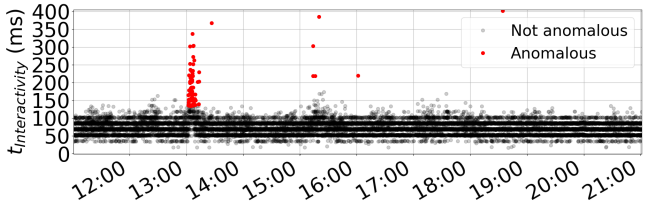

FIGURE 3: Example of labelling of active mode interactivity time series.

the represented anomalous samples. However, the available dataset exhibits certain peculiarities that limit the utility of such methods. The dataset contains contextual anomalies where the value of a sample alone may not determine its anomalous nature, as the figure 4 shows. Additionally, we observe the presence of background noise or non-anomalous periodic behaviours that can potentially be confused with anomalous patterns, emphasizing the importance of temporal context in our dataset. Given these data characteristics and a thorough review of the literature, we have chosen to employ recurrent neural networks, specifically LSTM, as they account for temporal, periodic, and repetitive anomalies, making them potentially useful for punctual, contextual, and collective anomalies.

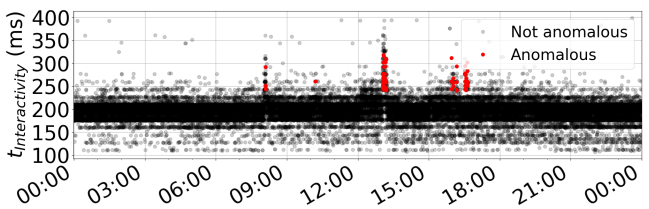

FIGURE 4: Example of labelling of a complex active mode interactivity time series

IV. INTERACTIVITY TIME ANOMALY DETECTION MODEL

A. MODEL STRUCTURE

In this section, we address the development of the machine learning model for anomaly detection in time series of interactivity time. We considered two main objectives in the design. Firstly, we prioritize the development of models that require a smaller number of features. The smaller the dimension of the feature space (fewer different features), the fewer observations are needed to cover the feature space. Additionally, observations of the same class become more compact. This phenomenon is known as the curse of dimensionality [47]. A smaller number of features requires a smaller volume of data, thus reducing the time required to generate a dataset for training the models. This shortens the time required for deploying the analysis tools in new environments. Secondly, we aim to develop a model that accurately detects anomalies in time series of interactivity time extracted in both active and passive modes.

The Figure 5 depicts the final structure of the machine learning model we employed for anomaly detection in time

series of interactivity time. The diagram does not show the internal details of the LSTM models. We developed this structure to be applicable to both types of time series: active mode and passive mode.

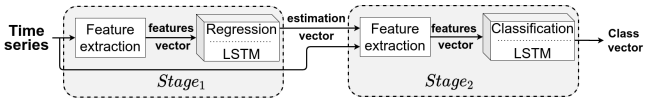

FIGURE 5: General diagram of the model for anomaly detection in time series of interactivity time.

The input to the general system is the time series of interactivity time, denoted as $t_{Interactivity}$. The structure consists of two stages. The purpose of the first stage is to perform a regression task, i.e., predicting values. Subsequently, the model uses this prediction to internally calculate the error by comparing it with the actual values. The error is then used to determine whether the samples are anomalous. The purpose of the second stage is to perform a classification task. In the case of the objective of anomaly detection in time series of interactivity time, it is a binary classification, indicating membership in the 'Anomalous' class (value 'True') or 'Non-anomalous' class (value 'False').

Figure 6 illustrates the detailed structure of the first stage. The figure does not show the internal details of the LSTM models. Stage 1 processes the time series x of interactivity time. This stage receives a window of N samples ($x[n+N]$) of interactivity time, and its objective is to estimate the average of the next temporal window with a shift of 1 sample (i.e., a window with a new sample). For each window of N samples of x, a new feature sample x' is generated. For each time instant n, the mean of all samples from n to $n+N$ is calculated to form a new sample of x'. The result is a vector of the mean of each interactivity measure and the previous N-1 samples. Next, this feature vector x' is standardized. Standardization is a common practice in machine learning techniques that facilitates algorithm convergence [48] [36]. It scales all features to a similar range so that, regardless of their original magnitude, they are comparable to each other. The result of this transformation is a new distribution of samples x'' with a mean $\mu = 0$ and standard deviation $\sigma = 1$. To achieve this, the mean of x' is subtracted and divided by the standard deviation of x'. The mean and standard deviation values must be calculated during training and used during evaluation.

The standardized feature vector x'' is the input to the LSTM network, which generates an output vector y. Each element i of the vector y represents the estimation of the next window with a shift of one sample ($\tilde{x}_{i+1}$). Due to the previous standardization, the result has a mean of 0 and a standard deviation of 1. To obtain the vector of estimations for the next windows (y'), the process needs to be reversed using the original μ and σ values that were stored earlier.

Figure 7 illustrates the detailed structure of the second stage. The figure does not show the internal details of the

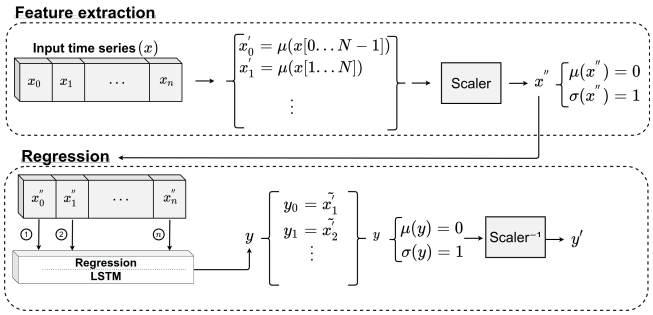

FIGURE 6: Diagram of stage 1 of the model for anomaly detection in interactivity time series.

LSTM models. Its objective is to generate an output vector o whose elements are binary values to estimate the class membership of the last sample in each interval.

The second stage receives as inputs the interactivity time samples x and the vector y' obtained from stage 1, referred to as 'prediction' from now on. Using these inputs, the feature matrix $\mathbf{F}$ is constructed, consisting of feature vectors f_j where $j \in [0, 5]$. The features employed in this second stage are:

- f_0: Prediction
- f_1: (Sample i) / Prediction
- f_2: Mean(window[i-N,i]) / Prediction
- f_3: Sample i
- f_4: Sample i / Mean (window[i-N,i])
- f_5: Sample i / Median(window[i-N,i])

Or expressed mathematically:

- $f_0 = y'$
- $f_1 = x/y'$
- $f_2 = \mu(x[i-N,i])/y'$
- $f_3 = x$
- $f_4 = x/\mu(x[i-N,i])$
- $f_5 = x/M_e(x[i-N,i])$

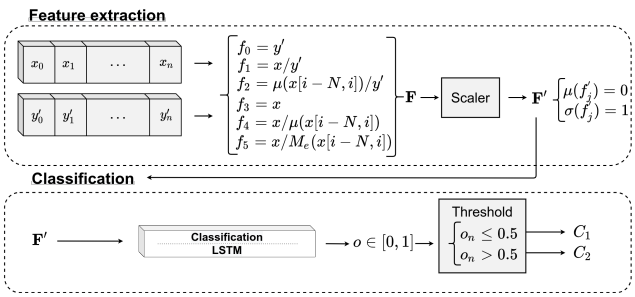

FIGURE 7: Diagram of stage 2 of the model for anomaly detection in interactivity time series.

Each vector of the feature matrix $\mathbf{F}$ should be standardized similarly to what was done in Stage 1. For each vector j of $\mathbf{F}$, the mean and standard deviation values from the model training must be stored for evaluation purposes. The result of this process is the matrix $\mathbf{F}'$. The new feature matrix serves as the input to the second LSTM network in the architecture, which is designed to generate an output vector

o with values ranging from 0 to 1. Each element of o is then subjected to a threshold (0.5) to determine its membership to a specific class. In the case of the anomaly detection objective in the time series of $t_{Interactivity}$, the classification is binary, indicating membership in the 'Anomalous' class (value > 0.5) or 'Non-anomalous' class (value $\leq$ 0.5).

We implemented the proposed structure using Python as the programming language. Specifically, we utilized the built-in functionalities of Python, numpy [49], and Pandas [50] for data management and processing. For the implementation of LSTM neural networks, we employed Scikit-learn [51] and Keras [52].

B. ADJUSTMENT OF MODEL PARAMETERS

In this section, we present the results obtained with the classification system, as well as the methodology used in the experiments. The experiments consist of four distinct parts: a) processing the data to adapt it to the input format of the network, b) randomly dividing the structured dataset into training and evaluation sets for cross-validation, c) constructing and training the models, and d) evaluating the system. The processes of division, training, and evaluation are repeated to ensure result consistency. Each repetition of this sequence is referred to as an iteration.

We repeat the processes of division, training, and evaluation multiple times using cross-validation to ensure result consistency. We evaluate the results of the different experiments and iterations using metrics commonly used in the field of machine learning: Accuracy, Precision, Recall (or sensitivity), and F1 score, defined in equations 1, 2, 3, 4, respectively. All these metrics provide output values ranging from 0 to 1.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1Score} = 2 \cdot \frac{\text{Precision Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

For all the available datasets, the model performs binary classification. In the case of detecting anomalous samples in time series of interactivity ($t_{Interactivity}$), Class C_1 corresponds to samples labelled as 'Anomalous'. This is the positive class, and its label value is 'True'. Lastly, Class C_2 is the complementary class and corresponds to samples identified as 'Non-Anomalous'. This is the negative class, and its label value is 'False'.

T_P is the number of true positives (samples that the model classify correctly as anomalous), F_P is the number of false positives (samples that the model classifies incorrectly as part of class C_1, anomalous samples), and F_N is the number of false negatives (samples that the model incorrectly classifies as the second class C_2, non-anomalous samples). The precision metric reflects the model's ability to correctly classify the total number of samples identified as positive (belonging to class C_1, anomalous samples). In other words, it reflects the tendency to classify samples as belonging to class C_1. On the other hand, the recall metric reflects the model's ability to correctly classify samples as belonging to class C_1 when they are part of that class. The F1 score metric is defined based on precision and recall and provides a way to quantify both metrics together.

Due to the nature of the dataset, we opted to maximize recall. We prefer the model to correctly classify as many samples as possible as 'Anomalous' in the time series of $t_{Interactivity}$. Maximizing recall over precision means preferring to identify a sample incorrectly as 'Anomalous' rather than labelling it as 'Non-anomalous'. This decision also aligns with the proportion of labelling in the available datasets. The dataset is highly imbalanced. We have a total of 106 samples out of 39,798 labelled as 'Anomalous' in the case of the dataset obtained in active mode, and a total of 165 samples out of 19,628 labelled as 'Anomalous' in the case of the dataset obtained in passive mode. Due to the small number of samples belonging to the classes of interest, we prefer the neural network to prioritize identifying them correctly and not opt for the opposite behaviour. This way, the model prioritizes the identification of the opposite class and making more errors with the 'Anomalous' samples.

The proposed model consists of two LSTM networks, one for each stage. Each network has two layers, an input layer, and an output layer. We use the rectified linear activation function (ReLU) for the regression task in the first stage and the hyperbolic tangent activation function (tanh) for the classification problem in the second stage. The number of neurons used in both stages is 20. The number of neurons affects the learning capacity of the network. Generally, more neurons can learn more complex patterns but may require longer training time. However, increased learning capacity can lead to overfitting, where the network easily learns the training data but struggles to generalize patterns and makes incorrect classifications or predictions on the evaluation data. We have experimentally adjusted the value of 20 neurons. Lower values showed limited learning capacity, while higher values resulted in unsatisfactory results on the evaluation dataset.

Both stages require an output layer. We use a layer called Dense, as described in the literature [53]. The purpose of this layer is to be fully connected to all neurons in the input layer. Depending on the machine learning problem, the Dense layer may have a different number of neurons. For regression classification, where the output value is only one, a single neuron with a linear activation function (commonly used for regression problems) is sufficient. For the second stage, which involves binary classification, the model could use two output neurons, where each activated output indicates membership in a specific class. However, it is possible to reduce the number of neurons in the Dense layer to one.

If the output is activated, it can be interpreted as belonging to one class, while its deactivation can be interpreted as belonging to the opposite class. This decision simplifies the implementation and reduces the computational complexity of training.

For both stages, we use the Adam optimizer [54] instead of the traditional stochastic gradient descent. Adam is computationally more efficient and has lower memory requirements. To promote convergence during training and prevent overfitting, we employ techniques such as Early Stopping [55] and Model Checkpoint [56]. These techniques help monitor the model's performance during training and stop the training process if no further improvement is observed or save the best model weights during training, respectively.

V. EVALUATION

A. EVALUATION OF ACTIVE MODE INTERACTIVITY TIME SERIES ANOMALY DETECTION

Figure 8 shows the evaluation metrics for different window sizes (N) that constitute the input to stage 1 of the model. The figure represents the results obtained by the model on the evaluation dataset, averaged over 100 iterations. The results obtained are satisfactory for any window size value. There is a slight improvement in the results as the window size increases. However, the results start to deteriorate from N=20 onwards. Therefore, for the final configuration, we choose this value as the optimal window size.

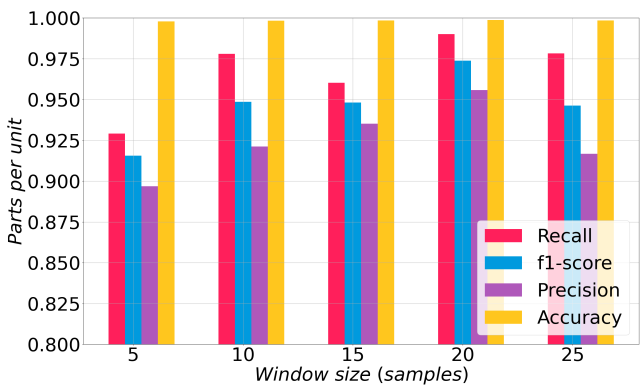

FIGURE 8: Average evaluation metrics for different window sizes in the $t_{Interactivity}$ time series anomaly detection model in active mode.

Figure 9 shows the classification result on the time series of the evaluation dataset for a window size of N=20. The classification result is positive. However, we observe the generation of false positives. To facilitate visualization, the figure 10 shows a temporal zoom on these intervals of time with anomalous samples

The figure 10 shows additional classification of anomalous samples. We collaborated with the service administrators of the remote desktops to identify the periods that the machine learning model should interpret as anomalous. However, these samples were not labelled as 'Anomalous' during the labelling phase, as they did not meet the administrators' criteria. In a real-world scenario, they could be considered anomalous. It is challenging for the analyst responsible for labelling the dataset to determine the boundary between an anomalous and a non-anomalous sample. The administrators did the labelling carefully; however, the network interprets that the samples represented in yellow are 'Anomalous' due to similarities. In a real implementation, this suggestion could be useful, and it would be up to the analyst to confirm whether it is a false positive or a true anomaly.

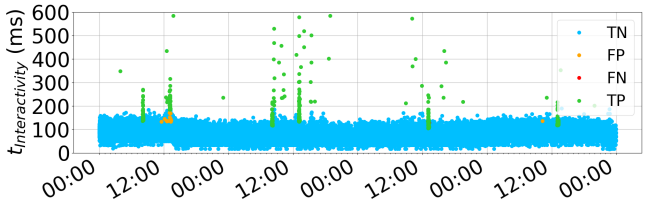

FIGURE 9: Classification result on the evaluation set of the time series of interactivity times obtained in active mode.

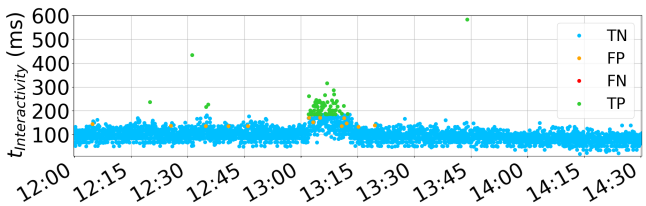

FIGURE 10: Zoom of the classification result on the evaluation set of the time series of interactivity times obtained in active mode.

Figure 11 shows the confusion matrix for the evaluation dataset, demonstrating the correct functioning of the architecture. The model misclassifies only 11 samples out of a total of 499 as 'Not anomalous'. In this scenario, the architecture is more prone to generating false positives due to the complexity of labelling, the decision to prioritize recall over precision, and the imbalance in the dataset. The false positive error amounts to 33 samples.

Finally, table 4 summarizes the evaluation metrics for the evaluation dataset using a window size of N=20.

TABLE 4: Summary of the evaluation metrics of the evaluation set for anomaly detection in $t_{Interactivity}$ time series obtained in active mode.

Accuracy	Precision	Recall	F1-Score
0.9997	0.9367	0.9779	0.9569

B. EVALUATION OF PASSIVE MODE TIME INTERACTIVITY TIME SERIES ANOMALY DETECTION

Passive mode interactivity time samples are not evenly spaced in time. The first stage performs a regression task, predicting the value of $t_{Interactivity}$ for the next window with a shift of one sample. When the samples are close in time (approximately 2 seconds), the prediction is easier as the transitions between sample values are smooth. However,

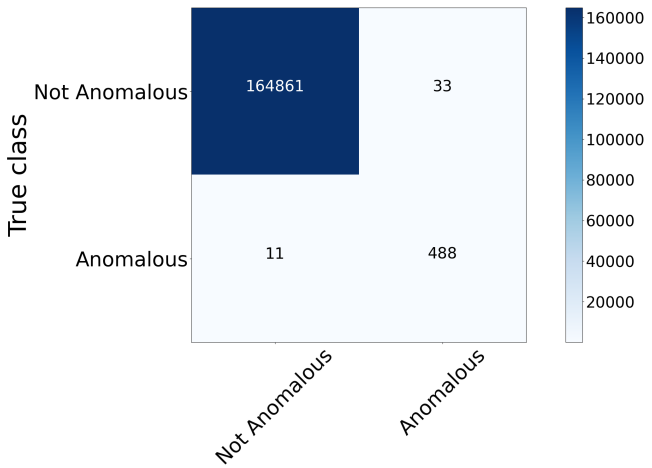

FIGURE 11: Confusion matrix of the evaluation set for anomaly detection in $t_{Interactivity}$ time series obtained in active mode.

in passive mode, the time interval between samples is determined by the user's actions. There can be a few milliseconds or minutes between two samples depending on the user's activity. For example, when editing a document, the time between samples may be determined by the typing frequency. But if the user is engaged in a less active task, such as reading a document, the frequency of clicks or keystrokes may decrease. In these scenarios, the prediction task becomes more challenging. Two distant timestamps may not have similar $t_{Interactivity}$ values as they are less correlated than two samples obtained within a few seconds. To address this issue, we artificially generate a dataset with a constant sample frequency using an interpolation process.

Starting from the beginning of the dataset, every T seconds, the stage 1 model introduces an interpolated sample based on the mean value of the $t_{Interactivity}$ samples from the previous T seconds. Figure 12 shows the new interpolated time series. The interpolated samples should be excluded when calculating the evaluation metrics. Only the classifications of real samples are taken into account. However, it is important to correctly label the added samples as the second stage of the architecture will learn from the available samples, including the interpolated ones. A sample is labelled as anomalous if the analysts label any of the passive measures within the sampling interval T as anomalous. Additionally, during periods of inactivity in remote desktops (holidays, nights, etc.), no interpolation was performed.

We performed a sweep of the interpolation interval T to determine the most suitable value for this variable. The figure 13 shows the sweep and the preference for an interpolation interval of 4 seconds. The figure shows a decreasing trend in the evaluation metrics as the value of the interpolation interval increases. The figure represents the results obtained by the model on the evaluation dataset, averaged over 100 iterations.

Figure 14 shows the evaluation metrics for different window sizes N that form the input to stage 1 of the model. The figure represents the results obtained by the model on the evaluation dataset averaged over 100 iterations. The results are satisfactory for any value of N. The sweep shows a positive trend as the window dimension increases. The best model achieves the best result with a sliding window size of N=20. Beyond this value, the results deteriorate.

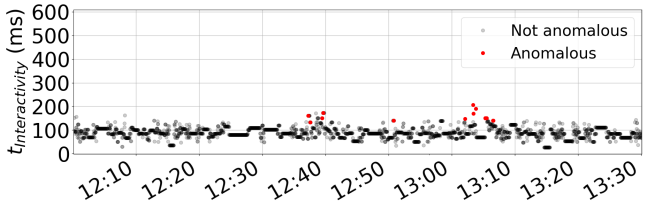

FIGURE 12: Zoom on the passive mode interactivity time series using an interpolation method.

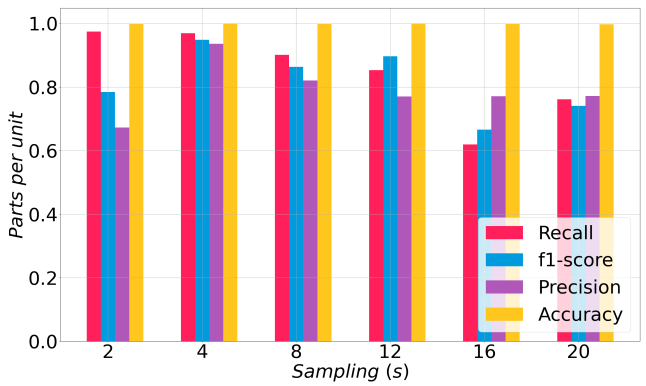

FIGURE 13: Average evaluation metrics for different temporal interpolation intervals in the anomaly detection model for passive mode time series of $t_{Interactivity}$.

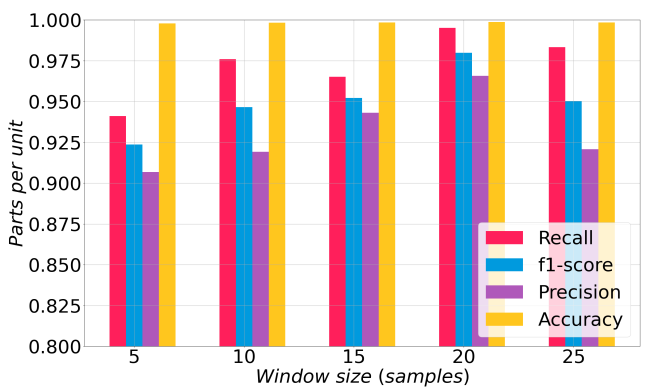

FIGURE 14: Average evaluation metrics for different window sizes in the anomaly detection model of $t_{Interactivity}$ time series in passive mode with a interpolation interval of 4 seconds.

Figure 15 shows the classification result on the time series of the evaluation dataset for a window size of N=20 with a 4-second interpolated data using the mean. The proposed model is able of generalizing the training data to the evaluation data. The figure 16 shows a temporal zoom of the same data where

we observe an absence of interpolated samples.

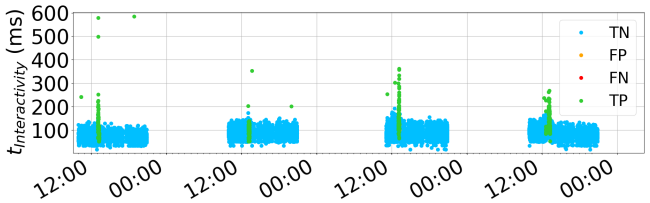

FIGURE 15: Classification result on the evaluation dataset of the time series of interactivity times obtained in passive mode using interpolation techniques.

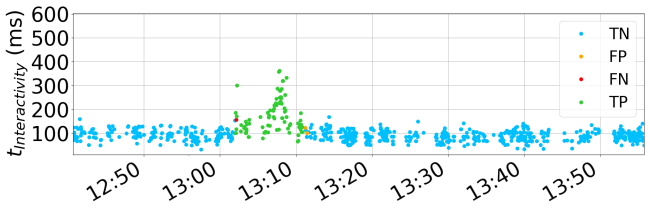

FIGURE 16: Temporal zoom of the classification result on the evaluation set of the time series of interactivity times obtained in passive mode using interpolation techniques.

Figure 17 shows the confusion matrix for the evaluation set and the correct functioning of the architecture. The model incorrectly classifies 19 samples as 'Not anomalous' out of a total of 474.

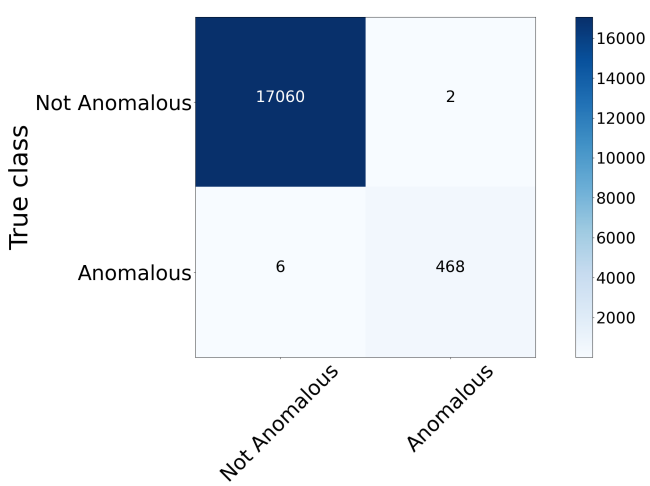

FIGURE 17: Confusion matrix for the evaluation dataset of anomaly detection in time series of $t_{Interactivity}$ obtained in passive mode using interpolation techniques.

Finally, the table 5 summarizes the evaluation metrics for the evaluation dataset using T = 4 s and N = 20.

TABLE 5: Summary of the evaluation metrics of the evaluation dataset for anomaly detection in $t_{Interactivity}$ time series obtained in passive mode and interpolation techniques.

Accuracy	Precision	Recall	F1-Score
0.9995	0.9957	0.9873	0.9915

Finally, there are hardly any differences in the results obtained by the model between the data obtained in active and passive modes. Despite the different measurement scenarios, the features we proposed allow for the elimination of the dependence on the environment, office, and users from which the interactivity measures are obtained. We demonstrated that thanks to interpolation, the results remain satisfactory in scenarios with data obtained in passive mode.

C. COMPARISON WITH OTHER ANOMALY DETECTION MODELS

To the best of our knowledge, no literature study has proposed an anomaly detection model for interactivity times. In this section, we compare the proposed model with other alternatives.

We used the dataset described in section 3.2 to compare the results of different models of anomaly detection. The Table 6 compiles the models that exhibited the best results. For comparison, we employed the same evaluation metrics described in the results of the LSTM model proposed in this article (Accuracy, Precision, Recall, F1-measure). Although all models display good Accuracy values, they encounter challenges with the Recall metric. While the evaluated models excel at determining when a sample is not anomalous, they struggle to identify anomalies satisfactorily. We attribute this phenomenon to the imbalanced nature of the data—there are very few anomalous samples compared to non-anomalous ones, posing a challenge for model learning. Moreover, unlike RNNs and specifically LSTM neural networks, these models do not consider the temporal information of interactivity time, resulting in the loss of valuable information for determining when a sample is anomalous. The results in Table 6 highlight the suitability of LSTM-based models for time series of interactivity time.

The results in Table 6 highlight the suitability of LSTM-based models for time series of interactivity time.

TABLE 6: Comparison with other anomaly detection models

	Model	Accuracy	Precision	Recall	F1-measure
Decision tree	Active	0.9047	0.0218	0.0715	0.0424
	Passive	0.9654	0.3716	0.5190	0.4331
Ensemble Decision forest	Active	0.9967	0.0673	0.0143	0.0235
	Passive	0.9711	0.4108	0.3059	0.3507
Ensemble Boosted Trees	Active	0.9971	0.2571	0.0183	0.0342
	Passive	0.9785	0.5841	0.5422	0.5624
Deep Neural Networks	Active	0.9869	0.6080	0.2546	0.0982
	Passive	0.9750	0.5140	0.3101	0.3868
Proposed LSTM model	Active	0.9997	0.9367	0.9779	0.9569
	Passive	0.9995	0.9957	0.9873	0.9915

VI. CONCLUSIONS

Interactivity time is the time the user perceives from when they interact with the application until they obtain a graphic

response. In this paper, we develop and implement a model for anomaly detection in time series of interactivity using machine learning strategies. We extensively analysed the state of the art of anomaly detection and to the best of our knowledge, no previous work targeted this scenario due to the novelty of the studied metric. Our proposal addresses a topic especially relevant in light of the prevailing remote work trend. For the development of the model, we leverage information available through a measurement tool based on TeCLA deployed in a university environment and a national parcel delivery company. We propose a single and novel architecture based on two LSTM networks that allows us to achieve the main objectives of the project: anomaly detection in time series of $t_{Interactivity}$ in both active and passive modes.

Our proposal approach successfully employs a reduced set of 6 features for the implementation of the machine learning model. We validate the model using over 165,504 samples in the active mode case and over 17,541 samples in the passive mode case. The model demonstrates satisfactory results for both active and passive operation modes. For anomaly detection in time series of $t_{Interactivity}$ in the active mode, the proposed model achieves an accuracy of 0.9997, precision of 0.9367, recall of 0.9779, and F1 score of 0.9569. In the case of anomalies in time series of $t_{Interactivity}$ in the passive mode, the model achieves an accuracy of 0.9995, precision of 0.9957, recall of 0.9873, and F1 score of 0.9915. These results clearly outperform other anomaly detection models that we have also analysed.

We bring to light crucial aspects that were previously unknown for the metric of interactivity time. We emphasize the importance of accurately labelling interactivity time datasets, fine-tuning model features through temporal windows, and underscore the need to interpolate time series in passive mode to achieve satisfactory model results.

As part of our future directions, this study delves into sample-by-sample anomaly detection, opening up avenues for refining the proposed model to analyse sample windows. Furthermore, recognizing the crucial role of accurate dataset labelling, we envision an exploration of unsupervised solutions as an important next step. The model we have developed has the potential to serve as a valuable tool for validating future implementations in this area.

REFERENCES

[1] H. Wang, R. Shea, X. Ma, F. Wang, and J. Liu, "On Design and Performance of Cloud-Based Distributed Interactive Applications," in 2014 IEEE 22nd International Conference on Network Protocols. Raleigh, NC, USA: IEEE, Oct. 2014, pp. 37–46.

[2] A. Menychtas, D. Kyriazis, S. Gogouvitis, K. Oberle, T. Voith, G. Galizo, S. Berger, E. Oliveros, and M. Boniface, "A cloud platform for real-time interactive applications," p. 7, Feb. 2011.

[3] A. Bartik, Z. Cullen, E. L. Glaeser, M. Luca, and C. Stanton, "What Jobs are Being Done at Home During the COVID-19 Crisis? Evidence from Firm-Level Surveys," SSRN Electronic Journal, 2020.

[4] C. Carraher-Wolverton, "The co-evolution of remote work and expectations in a COVID-19 world utilizing an expectation disconfirmation theory lens," Journal of Systems and Information Technology, vol. 24, no. 1, pp. 55–69, Feb. 2022.

[5] "COVID-19and remote work: An early look at US data," https://www.brynjolfsson.com/remotework.

[6] Teradici, "Remote Work 2020 Report - The Separation Of Work And Place," https://connect.teradici.com/remote-work-2020.

[7] Teradici, "Remote Work 2021 Report - The Separation Of Work And Place," https://connect.teradici.com/remote-work-2021.

[8] Markets and Markets, "Cloud Gaming Market by Offering (Infrastructure, Gaming Platform Services), Device Type (Smartphones, Tablets, Gaming Consoles, PCs & Laptops, Smart TVs, HMDs), Solution (Video Streaming, File Streaming), Gamer Type, Region - Global Forecast to 2024," 2021.

[9] A. Richter, "Locked-down digital work," International Journal of Information Management, vol. 55, p. 102157, Dec. 2020.

[10] Y. A. Wang, C. Huang, J. Li, and K. W. Ross, "Estimating the performance of hypothetical cloud service deployments: A measurement-based approach," in 2011 Proceedings IEEE INFOCOM. Shanghai, China: IEEE, Apr. 2011, pp. 2372–2380.

[11] Y. Yeboah Junior and X. Hei, "Evaluating the performance of cloud services in a browser-based network measurement platform," in 2013 19th IEEE International Conference on Networks (ICON). Singapore: IEEE, Dec. 2013, pp. 1–6.

[12] W. Wu and D. Shang, "Employee Usage Intention of Ubiquitous Learning Technology: An Integrative View of User Perception Regarding Interactivity, Software, and Hardware," IEEE Access, vol. 7, pp. 34 170–34 178, 2019.

[13] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," IEEE Communications Magazine, vol. 50, no. 4, pp. 28–36, Apr. 2012.

[14] J. Arellano-Uson, E. Magaña, D. Morató, and M. Izal, "Protocol-agnostic method for monitoring interactivity time in remote desktop services," Multimedia Tools and Applications, Feb. 2021.

[15] F. Safaei, P. Boustead, C. Nguyen, J. Brun, and M. Dowlatshahi, "Latency-driven distribution: Infrastructure needs of participatory entertainment applications," IEEE Communications Magazine, vol. 43, no. 5, pp. 106–112, May 2005.

[16] V. M. G. Abarca, P. R. Palos-Sanchez, and E. Rus-Arias, "Working in Virtual Teams: A Systematic Literature Review and a Bibliometric Analysis," IEEE Access, vol. 8, pp. 168 923–168 940, 2020.

[17] J. Nieh, S. J. Yang, and N. Novik, "Measuring thin-client performance using slow-motion benchmarking," ACM Transactions on Computer Systems, vol. 21, no. 1, pp. 87–115, Feb. 2003.

[18] A. M. Lai and J. Nieh, "On the performance of wide-area thin-client computing," ACM Transactions on Computer Systems, vol. 24, no. 2, pp. 175–209, May 2006.

[19] A. Berryman, P. Calyam, M. Honigford, and A. M. Lai, "VDBench: A Benchmarking Toolkit for Thin-Client Based Virtual Desktop Environments," in 2010 IEEE Second International Conference on Cloud Computing Technology and Science. IEEE, Nov. 2010, pp. 480–487.

[20] T. Nguyen, P. Calyam, and R. B. Antequera, "Benchmarking in virtual desktops for end-to-end performance traceability," in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). Ottawa, ON, Canada: IEEE, May 2015, pp. 1268–1273.

[21] F. Alali, T. A. Adams, R. W. Foley, D. Kilper, R. D. Williams, and M. Veeraraghavan, "Methods for Objective and Subjective Evaluation of Zero-Client Computing," IEEE Access, vol. 7, pp. 94 569–94 582, 2019.

[22] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.

[23] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[24] M. Braei and S. Wagner, "Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art," Apr. 2020.

[25] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol. 60, pp. 19–31, Jan. 2016.

[26] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," Signal Processing, vol. 99, pp. 215–249, Jun. 2014.

[27] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data: A Survey," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[28] J. C. B. Gamboa, "Deep Learning for Time-Series Analysis," 2017.

[29] W. F. Guthrie, "NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151)," 2020.

[30] J. Shi, G. He, and X. Liu, "Anomaly Detection for Key Performance Indicators Through Machine Learning," in 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC). Guiyang: IEEE, Aug. 2018, pp. 1–5.

[31] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," WIREs Data Mining and Knowledge Discovery, vol. 1, no. 1, pp. 73–79, Jan. 2011.

[32] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks," in Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham: Springer International Publishing, 2019, vol. 11730, pp. 703–716.

[33] J. Struye and S. Latré, "Hierarchical temporal memory and recurrent neural networks for time series prediction: An empirical validation and reduction to multilayer perceptrons," Neurocomputing, vol. 396, pp. 291–301, Jul. 2020.

[34] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data: A Survey," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[35] C. C. Aggarwal et al., "Neural networks and deep learning," Springer, vol. 10, no. 978, p. 3, 2018.

[36] K. Kawakami, "Supervised sequence labelling with recurrent neural networks," Ph.D. dissertation, Technical University of Munich, 2008.

[37] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[38] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Anchorage AK USA: ACM, Jul. 2019, pp. 2828–2837.

[39] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ser. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2712–2721.

[40] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," Neurocomputing, vol. 262, pp. 134–147, Nov. 2017.

[41] H. Xu, Y. Feng, J. Chen, Z. Wang, H. Qiao, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, and D. Pei, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications," in Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18. Lyon, France: ACM Press, 2018, pp. 187–196.

[42] D. Perdices, J. E. L. De Vergara, and J. Ramos, "Deep-FDA: Using Functional Data Analysis and Neural Networks to Characterize Network Services Time Series," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 986–999, Mar. 2021.

[43] Markets and Markets, "Deep Learning Market by Offering (Hardware, Software, and Services), Application (Image Recognition, Signal Recognition, Data Mining), End-User Industry (Security, Marketing, Healthcare, Fintech, Automotive, Law), and Geography - Global Forecast to 2023," https://secure.livechatinc.com/.

[44] X. Su, M. Wu, and J. Xu, "A novel virtual storage area network solution for virtual desktop infrastructure," in 2014 International Symposium on Wireless Personal Multimedia Communications (WPMC). Sydney, Australia: IEEE, Sep. 2014, pp. 204–208.

[45] "Citrix - All in one Workspace Solution for Secure Access," https://www.citrix.com/.

[46] "Windows Remote Desktop - Remote Desktop clients for Remote Desktop Services and remote PCs," https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients.

[47] L. Chen, "Curse of Dimensionality," in Encyclopedia of Database Systems, L. Liu and M. T. Özsu, Eds. Boston, MA: Springer US, 2009, pp. 545–546.

[48] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in Neural Networks: Tricks of the Trade. Springer, 2002, pp. 9–50.

[49] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[50] W. McKinney, "Data Structures for Statistical Computing in Python," in Python in Science Conference, Austin, Texas, 2010, pp. 56–61.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[52] F. Chollet et al., "Keras," 2015.

[53] V. H. Josephine, AP. Nirmala, and V. L. Alluri, "Impact of hidden dense layers in convolutional neural network to enhance performance of classification model," in IOP Conference Series: Materials Science and Engineering, vol. 1131. IOP Publishing, 2021, p. 012007.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015.

[55] Y. Yao, L. Rosasco, and A. Caponnetto, "On Early Stopping in Gradient Descent Learning," Constructive Approximation, vol. 26, no. 2, pp. 289–315, Aug. 2007.

[56] F. Wang, G. Wei, Q. Liu, J. Ou, H. Lv et al., "Boost neural networks by checkpoints," Advances in Neural Information Processing Systems, vol. 34, pp. 19 719–19 729, 2021.

JESUS ARELLANO-USON graduated in telecommunication engineering from the Public University of Navarre (UPNA), Spain, in 2019. He received the M.Sc. degree in telecommunication engineering from UPNA, in 2021, where he is currently pursuing the Ph.D. degree with the Telecommunications, Networks and Services Research Group. Previously, in 2018, he held a Scholarship at the Automatics and Computing Department. In 2019, 2020 and 2021, he was a Research Assistant with the Telecommunications, Networks and Services Research Group, UPNA.

EDUARDO MAGAÑA received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Public University of Navarre, Pamplona, Spain, in 1998 and 2001, respectively. Since 2005, he has been an Associate Professor with the Public University of Navarra. In 2002, he was a Postdoctoral Visiting Research Fellow with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley. His main research interests are network monitoring, traffic analysis, and performance evaluation of communication networks.

DANIEL MORATO received the M.Sc. degree in telecommunication engineering and the Ph.D. degree from the Public University of Navarre, Spain. In 2002, he was a Visiting Postdoctoral Fellow with the Electrical Engineering and Computer Sciences Department, University of California at Berkeley, Berkeley. Since 2006, he has been working at the Public University of Navarre. He is currently an Associate Professor with the Department of Electrical, Electronic and Communications Engineering. In 2014, he became a member of the Institute of Smart Cities. His research interests include high-speed networks, the performance and traffic analysis of Internet services, and network monitoring.

MIKEL IZAL received the M.Sc. and Ph.D. degrees in telecommunication engineering, in 1997 and 2002, respectively. In 2003, he worked as a Scientific Visitant at the Institute Eurecom, France, performing measures in network tomography and peer-to-peer systems. Since 2013, he has been with the Public University of Navarre, where he is currently an Associate Professor. His research interests include traffic analysis, network tomography, high-speed next generation networks, and peer-to-peer systems.