Multi-signal Anomaly Detection for Real-Time Embedded Systems

by

Reinier Torres Labrada

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents MuSADET, an anomaly detection framework targeting timing anomalies found in event traces from real-time embedded systems. The method leverages stationary event generators, signal processing, and distance metrics to classify inter-arrival time sequences as normal/anomalous. Experimental evaluation of traces collected from two real-time embedded systems provides empirical evidence of MuSADET's anomaly detection performance.

MuSADET is appropriate for embedded systems, where many event generators are intrinsically recurrent and generate stationary sequences of timestamp. To find timing-anomalies, MuSADET compares the frequency domain features of an unknown trace to a normal model trained from well-behaved executions of the system. Each signal in the analysis trace receives a normal/anomalous score, which can help engineers isolate the source of the anomaly.

Empirical evidence of anomaly detection performed on traces collected from an industry-grade hexacopter and the Controller Area Network (CAN) bus deployed in a real vehicle demonstrates the feasibility of the proposed method. In all case studies, anomaly detection did not require an anomaly model while achieving high detection rates. For some of the studied scenarios, the true positive detection rate goes above 99 %, with false-positive rates below one %. The visualization of classification scores shows that some timing anomalies can propagate to multiple signals within the system. Comparison to the similar method, Signal Processing for Trace Analysis (SiPTA), indicates that MuSADET is superior in detection performance and provides complementary information that can help link anomalies to the process where they occurred.

Acknowledgements

I would like to thank all the people who made this thesis possible. Especially Professor Sebastian Fischmeister for his patience and support.

Especial thanks to Professor Zhou Wang, and Professor Rodolfo Pellizzoni for their excelent feedback and dedication to review this thesis. Your input helped me improve the quality of this work.

Dedication

This is dedicated to my wife and daughter for their love an patience during all the missing hours.

Table of Contents

List of Figures

List of Tables

Abbreviations

AD anomaly detection 9, 67

BPSS binary power spectral sequence vii, 6, 37–39, 42, 43, 60–63

CAN controller area network 7, 21, 28, 32, 45, 48–50, 52–58, 67

DC direct current 6, 23, 28, 29, 32–34, 40, 43, 44, 53, 58, 67

DCTPR DC to total power ratio vii, 6, 32–34, 43, 47, 50, 57, 60, 67

DFT Discrete Fourier Transform 33

DOS denial of service 8

DTFT Discrete Time Fourier Transform 12

EDF earliest-deadline first 10

ETP execution-time profile 3

FD frequency domain 5, 10, 15, 16, 20, 34, 37–40, 62, 67

FT Fourier Transform 9, 35, 37

IATS inter-arrival times sequence vii, 3–6, 9, 10, 12, 13, 15, 20–24, 27–29, 31–37, 39, 40, 43, 44, 52, 53, 55, 58, 60, 62, 67

IID independent identically distributed 27, 28

MSE mean square error 36

MuSADET multi-signal anomaly detection for real-time traces ix, 2, 4–7, 9, 11–13, 15, 16, 18–23, 36, 39, 42, 45–47, 50–54, 56, 58, 60–63, 66, 67

PCP priority ceiling protocol 11

PSD power spectral density vii, viii, 5, 6, 12, 13, 15, 16, 18, 27–29, 32, 34–40, 42, 60–64, 66, 67

QNX QNX operating system 58

QoS quality of service 8

RM rate monotonic 10

ROC receiver operating characteristic curve 46, 47, 52–54, 58, 59, 61, 63, 66

RTA response-time analysis 10, 11

RTES real-time embedded system 1–3, 10, 11

RTS real-time system 9–11, 27

SiPTA Signal Processing for Trace Analysis 5, 6, 9, 45–47, 53, 58, 60–62, 66, 67

TSS time-stamp sequence 27

WCET worst-case execution time 3

WSS wide-sense stationary 5, 9, 11, 12, 28, 29, 32, 34, 40

Chapter 1

Introduction

Real-time embedded systems (RTESs) control most of today's sophisticated and technologically advanced infrastructure. Undetected defects or malfunctions of the controlling devices can lead to catastrophic and costly failures such as the Therac-25 [54] medical particle accelerator, and the Ariane 5 flight [47]. As RTES become more complex their safety requirements turn harder to assess [41]. Approaching the issue of deploying safe systems requires the use of tools capable of detecting problems or potentially problematic conditions before they pose a threat.

There are several tools applicable to RTES and oriented toward meeting safety requirements. For example, real-time systems theory analyzes the system under its worst-case timing scenario to check response-time deadlines. Model-checking formally verifies a model of the system to check if it meets a given specification. Runtime verification checks the output of the running system towards a formal specification. Testing verifies the output of the system run under a particular set of scenarios. Anomaly detection looks for outliers in the observable output of the system that may be indicators of unsafe conditions. None of these techniques alone can guarantee that the system will behave safely upon deployment. Real-time theory cannot provide guarantees on the system's input/output correctness. Due to model state-explosion, model-checking tends to be computationally unfeasible. Runtime verification depends on the completeness and accuracy of the system's formal specification. Testing cannot cover the whole input/output space of the system, and anomaly detection may rise false alarms or may fail to detect hazardous conditions.

In this thesis, we focus on monitoring employing anomaly detection to tackle one facet of the unsolved problem of incomplete verification for complex embedded software. For most systems, it is impossible to explore all possible values of its input/output domain during testing, and as a result, monitoring is still the ad hoc engineering solution for error detection [51]. Monitoring is, therefore, a requirement in safety standards for embedded systems, such as ISO-26262 for functional automotive [43], and DO-178C for airborne systems [36]. Monitoring aims to track that part of the input/output domain visited by the system during its operation to check whether or not it meets the specification. Among the many different monitoring mechanisms, anomaly detection tries to detect uncommon behaviour or bursts of unexpected activity [19].

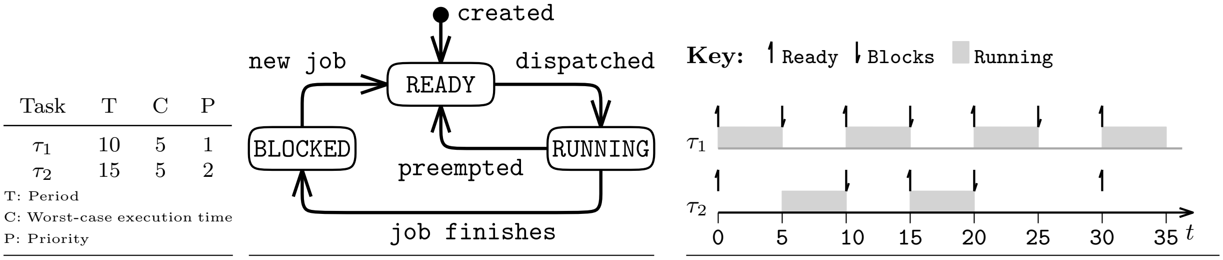
The difficulty in detecting anomalies is twofold: First, the system output for anomalous behaviour can be in the range of the input/output domain, and second, anomalies tend to be single occurrences of unexpected behaviour [20]. Moreover, anomalies are not necessarily an indication of a failure or unsafe condition; they are an unknown and infrequent system state that needs analysis. In the case of RTES, timing anomalies are of great importance because they can lead to missed deadlines or may be a symptom of malfunction.

1.1 Motivating example

Multi-signal anomaly detection for real-time traces (MuSADET) integrates real-time systems, signal processing and anomaly detection. The following example is an attempt to familiarize the reader with the topic.

Consider a real-time system implemented as a task set of two tasks with task properties shown in Figure 1.1.a. Each task can change state according to the transition diagram of Figure 1.1.b. This system will produce the scheduling plot of Figure 1.1.c. Both tasks are periodic and scheduled by a rate monotonic scheduler [53, 32]. Either task can change its state by itself or due to scheduling actions. For example, any task will become suspended at the end of a job (self-change), while τ_2 is preempted soon after τ_1 becomes ready (scheduling-action). Whenever a task changes state, the operating system issues a new trace entry for that specific event.

The schedule in Figure 1.1.c repeats starting at $t = 30$ producing a periodic sequence of events. This deterministic task model simplifies time correctness analysis [51] and is the primary motivation behind a large research body under the real-time system theory [71]. Real-time scheduling is attractive because it analyzes a deterministic case with polynomial or pseudo-polynomial time complexity, while general scheduling is known to have NP-Hard complexity [22]. As a result, many RTES are implemented as periodic task sets to simplify timing analysis with guarantees of meeting the systems deadlines. Hence, a periodic and well-behaved task set should generate events that exhibit recurrent behaviour. In practice,



Task	T	C	P
τ_1	10	5	1
τ_2	15	5	2

T: Period
C: Worst-case execution time
P: Priority

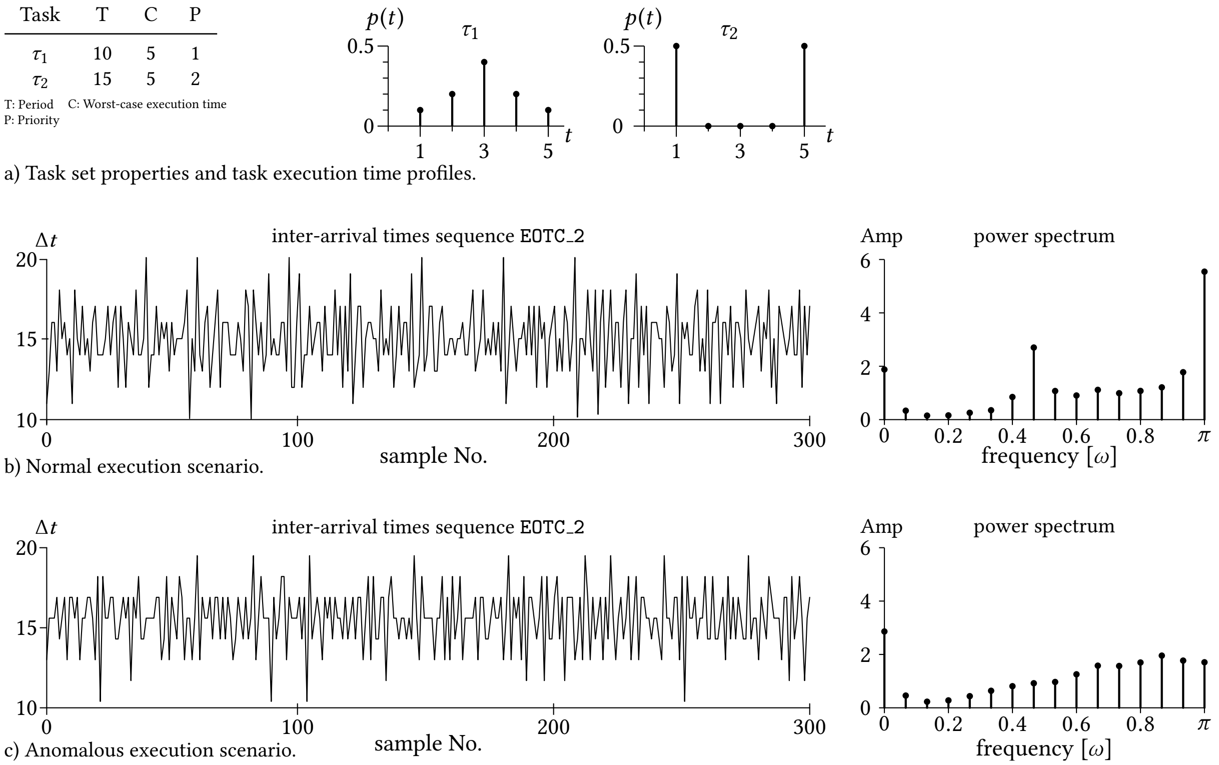
a) Task set properties. b) Task's transition diagram. c) Schedule.

Figure 1.1: A system of two periodic tasks with deterministic execution times. The system is defined by the properties in table (a). The transition state diagram shown in (b) will produce the periodic schedule shown in (c). The schedule repeats after t=30 due to the use of worst-case execution times.

due to scheduling actions and tasks execution-time profiles (ETPs) the timestamps of events generated by the system should be considered non-deterministic.

One could assume that a feasible periodic task set would not be affected by timing issues that may cause abnormal behaviour. However, according to [77], timing anomalies can induce chaotic behaviour. We argue that even when timing-anomalies do not bring instability to a system, their effects are detectable through trace analysis. For example, Figure 1.2 presents the inter-arrival times sequence (IATS) for the `Job Completes (JC)` event of τ_2 for the deterministic system of Figure 1.1. Instead of worst-case execution times (WCETs), we used ETPs when simulating tasks. Due to the use of ETPs, both tasks generate their events at random timestamps. The periodic activation of τ_1 and τ_2 guarantees a constant rate of events per hyper-period, while the ETP randomizes events timestamps. The plot for the IATS and its corresponding power spectrum in Figure 1.2.b being the expected behaviour. Swapping the priorities of τ_1 and τ_2 preserves the system's feasibility, but violates tasks precedencies as shown in Figure 1.2.c. By inspecting the IATS in Figure 1.2, we would conclude that there is no apparent difference between the normal and anomalous IATS while the power spectra are distinctively different. If, for example, an attacker swaps task's priorities, the priority swap can be detected by finding the similarity between the power spectra of normal and anomalous IATS and then assigning an anomaly score employing a classification algorithm. This example shows the potential for detecting timing anomalies by using signal processing to extract frequency domain features from IATS. To complete the process, we will require methods to compare features effectively, and a method to decide whether the system is behaving normally or anomalously.

Since recurrent processes are at the foundation of RTES, we could use the IATS gen-



Task	T	C	P
τ_1	10	5	1
τ_2	15	5	2

T: Period C: Worst-case execution time
P: Priority

a) Task set properties and task execution time profiles.

b) Normal execution scenario.

c) Anomalous execution scenario.

Figure 1.2: A system of two tasks showing normal/anomalous behaviour. The Cheddar [72] simulated task-set in (a) produces the normal IATS for τ_2 JOB COMPLETES (JC) event in (b). Swapping tasks priorities for several jobs of τ_2 generates anomalous IATS shown in (c). The anomalous behaviour is detectable by comparing the IATS power spectra.

erated by such processes to learn features from well-behaved executions of the system. Detecting timing anomalies would be accomplished by comparing the learned features to features extracted from traces recorded under unknown operating conditions. Using well-behaved executions to train a normal model allows considering the system under analysis as a black-box system. The black-box abstraction enables anomaly detection without the need for an anomaly model. Not requiring an anomaly model allows MuSADET to perform anomaly-detection on any system with recurrent generators, and a tracing mechanism that records timestamps and identifiers for generators and signals.

1.2 Problem statement

Given a model of normal behaviour learned from well-behaved executions of the system, and some test trace recorded under unknown conditions automatically determine whether or not there are timing anomalies in the test trace. For each anomaly found, report the signals and time windows in which the anomalies are present.

We loosely define a timing anomaly as a *sequence of inter-arrival times whose frequency domain features are dissimilar to features in the normal model*. We assume that IATS are at least wide-sense stationary (WSS) and, thus, will have well-defined frequency domain (FD) features. Since IATS can be long, we divide it into windows, each covering some time interval. MuSADET assigns a normal/anomalous score to each signal in the window. Therefore, due to the nature of data and choice of feature, our method is appropriate for scenarios where anomalous behaviour is persistent during the time-span of the window under analysis.

The problem statement defines a semi-supervised binary classification problem. MuSADET requires traces considered normal to build a model. Then test traces can be compared to check if they conform to the learned model. The main goal of using normal traces to construct a model of the system is the minimization of false positives and false negatives error rates. We introduce the framework for anomaly detection in Chapter 3.

1.3 Contributions

The main contributions of this work are as follows:

Trace and signal modelling

Our method models signals as renewal Poisson processes. First, we relax the stationarity requirement of IATS to WSS. Second, we apply nonparametric power spectral density (PSD) estimation to compute the power spectrum from the signal's IATS. Our method improves over [85] by solving issues in its trace to signal model. The fundamental issue of Signal Processing for Trace Analysis (SiPTA) is that it can extract sequences with duplicate timestamps. Sequences with duplicate timestamps result from joining at least two nonindependent renewal processes, which is a violation of the superposition principle [25].

Features

We show that an IATS can be (i) a constant amplitude signal or (ii) a wide sense stationary sequence. We propose two mutually exclusive features to classify IATS as normal/anomalous. The DC to total power ratio (DCTPR) can be used to classify only direct current (DC) IATS, while PSD estimates are better suited for stationary IATS. PSD estimation methods are among the oldest in signal theory. However, its use for timing anomaly detection on event traces still provides new insights and needs further exploration. We present an improvement of the feature proposed in [85] by using Welch periodograms [81, 61], and by separating DC from non-DC IATS.

Augmented anomaly detection granularity

We propose a technique that improves over [85]. Instead of assigning a score to the whole trace, we classify signals independently and allow adjusting the length of the classification window. Anomaly detection at the signal level facilitates the isolation of timing anomalies. Being able to use different window lengths enables analysis at different time bands, which in turn determines the response time of the detector when used in an online environment.

Improvement of classification performance

We tested MuSADET with two datasets, and demonstrate with the HCRL-CAN dataset that SiPTA [85] fails when a system is mainly composed of generators producing DC IATS. For the HEXACOPTER dataset, we show that although SiPTA still performs well, MuSADET provides better accuracy and robustness. We propose the χ^2 symmetric distance [14] based on Pearson and Neyman's distances [68] to compare PSDs, which includes the whole spectrum versus the single peak method of [85]. Finally, we tested the Jaccard similarity [30] on the binary power spectral sequence (BPSS) derived from the PSD, also with better performance than SiPTA.

1.4 Thesis Organization

We begin Chapter 2 with a background review of anomaly detection applied to scenarios similar to those we target with MuSADET. Chapter 2 continues with an introduction to the fundamental concepts supporting MuSADET and finishes with a general overview of anomaly detection. We present MuSADET's framework in Chapter 3 along with its underlying mathematical model for trace parsing into signals, feature extraction from said signals and classification of test features. Chapter 4 presents and discusses two case studies where we applied MuSADET to detect anomalies. The first case tackles intrusion detection in the controller area network (CAN) bus system of a commercial grade car. The second case addresses timing anomalies in a HEXACOPTER where the system is disturbed by a non-terminating process or excessive use of the input-output subsystem. Finally, we present our conclusions in Chapter 5.

Chapter 2

Related Work and Background

2.1 Related Work

Finding anomalies soon after they occur is almost impossible. Timing anomalies, in particular, need progression over time to achieve detection. For example, denial of service (DOS) attacks is one well-studied problem where instantaneous detection is impossible [49, 1, 62]. Among DOS attacks, low rate stealthy DOS modes [21, 56, 1] are harder to detect because they do not flood the victim with network packets, and therefore the attack must persist for some time before the detector can raise the alarm. Intrusion detection in information systems [84], metamorphic virus detection [78], and temporal debugging of real-time applications with quality of service (QoS) requirements [39] also demand some level of progression or presence of abnormal sequences to achieve detection.

The type of data, its generating process and the technique used to find anomalies affect the anomaly detection performance on sequence data. Moreover, data can be processed in its native domain or after applying some transformation [16, 17].

Markov models are suitable for sequences of event symbols and can detect anomalies within long sequences but suffer from high false-negative rates [16]. For example, Markov techniques are used in [83, 60] to learn a model from normal sequences and then use the learned model to predict the probability of observing each symbol of the test sequence. Hidden Markov Models (HMM) [34] techniques learn a model from normal sequences and then compute the probability of test sequences being normal/anomalous. Markov modelling methods are useful for sequences of random symbols (e.g., user identifiers or event names) that would not benefit from domain transformation [16].

Similarity-based anomaly detection (AD), is simple and highly effective when one of the sequences has a defined pattern [17]. For example, similarity-based techniques [50, 8] are applied to sequences of fixed length, resulting from aggregations of random variables or presence/absence of features [52, 5]. Distance-based methods are commonly applied to vectors that represent points in Euclidean space or are the result of a domain transformation of a base sequence vector (usually from time-domain to frequency-domain). Some distance/similarities are equivalent, depending on the mathematical model used [68].

Signal processing is a standard tool used for domain transformation of base sequences. For instance, in [35, 23, 67, 65, 58], wavelet transforms were applied to anomaly detection in network systems while the Fourier transform was used in [86]. In most cases, AD performed using signal processing focus on signals derived by counting packets [21, 1] in the network. In contrast, multi-signal anomaly detection for real-time traces (MuSADET) focuses on the inter-arrival times of consecutive events for some well-defined class of events.

In [85], Mehdi et al. presented Signal Processing for Trace Analysis (SiPTA), an anomaly detection engine for embedded systems that demonstrated the feasibility of using signal processing and inter-arrival times sequence (IATS) to analyze event traces. Despite its promising results, SiPTA relies on incorrect assumptions. For example, SiPTA applies the Fourier Transform (FT) to extract the frequency spectrum of time-stamp sequences and then records the component of maximum amplitude and the frequency where it occurred. This approach assumes that IATS are periodic sequences, and we showed in Figure 1.2 that IATS could be non-periodic. Also, using a single peak for classification of a set of IATS assumes the existence of a single dominant peak located around the same frequency. Lastly, the normalization applied to the amplitude spectrum requires the same length for model and test sequences limiting the application of SiPTA to a single detection window. MuSADET borrows some principles from [85] but applies a different classification strategy that improves anomaly detection accuracy and tackles all previously mentioned shortcomings.

2.2 Background

2.2.1 Real-Time Systems Theory

For MuSADET, the Real-time system (RTS) theory provides the framework sustaining the fundamental assumption that signal generators are wide-sense stationary (WSS) processes. In our motivating example presented in Section 1.1, we showed that periodic tasks generate events in a recurrent fashion that can be modelled by their frequency domain (FD) features. RTS theory heavily relies on the notion of periodic tasks or aperiodic tasks modelled as periodic or sporadic equivalents. Hence real-time embedded systems (RTESs) are commonly implemented under the paradigm of periodic task sets.

The central motivation behind the real-time systems theory is to determine if a given task set will meet its deadlines under the worst-case execution scenario. According to [32, 22, 71], theoretical results suitable to be applied in real applications exist since the seminal paper of Liu and Layland [55], which settled the foundations for schedulability analysis based on the processor utilization factor. The basic model comprises a task set $\Gamma_n \triangleq \langle \tau_1, \tau_2, \ldots, \tau_n \rangle$ with n periodic and independent tasks, each task τ_i having period T_i, and worst-case execution time or capacity C_i. Task set Γ_n is guaranteed to be feasible when scheduled by the Rate Monotonic algorithm if the following relation holds:

$$U_{tot} \leq n \left(2^{1/n} - 1 \right) \tag{2.1}$$

Where the total processor utilization factor is:

$$U_{tot} = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2.2}$$

A result of this model is that the schedule repeats after the hyper-period $T_h \triangleq \mathrm{lcm}\{T_i\}, i = 1, 2, \ldots, n$, the least common multiple of all periods. We have seen from Figure 1.1 that tasks executed with deterministic computation times will produce periodic IATS.

Even if Liu and Layland's model only accept independent periodic tasks, they proved [55] that rate monotonic (RM), and earliest-deadline first (EDF) are optimal scheduling algorithms for static and dynamic priorities assignments respectively [53, 32]. The result shown in (2.1) is relevant because the rate monotonic algorithm can be implemented over fixed-priority preemptive schedulers.

Response-time analysis (RTA), introduced by Joseph and Pandya in 1986 [42], is another important branch in the RTS theory. Rather than computing the processing load of each task, this approach relies on calculating the time a task needs to finish a released job. Response-time analysis can compute the load, and the interference suffered due to preemption from tasks with higher priorities to the task under analysis. The basic test can be solved by the following recurrent relation [2]:

$$R_i^{n+1} = \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j \tag{2.3}$$

Where R_i^{n+1} is the response-time of the current iteration, the test stops with the response-time R_i of task τ_i if $R_i^{n+1} = R_i^n$ and $R_i^{n+1} \leq D_i$ or fails if at any point $R_i^{n+1} > D_i$.

There are two notable implications of applying RTA to RTES. First, the hyper period principle is preserved, hence under the deterministic worst-case model; the system schedule becomes periodic with the period T_h. Second, RTA can incorporate blocking and delays that prevent the task to gain access to the processor, and also tasks with deadlines shorter than periods [32, 71].

Further developments in real-time theory included tests for systems with tasks that share resources or suffer release jitter. The analysis requires the use of the Priority ceiling protocol (PCP) to avoid deadlock and cope with priority inversion. There are equations for systems with sporadically repeating tasks, a mention to tick scheduling, and precedence relations between cooperating tasks that run on different processors. Burns et al. [11, 12] tackled the analysis of tick driven schedulers that employ lists to hold the ready and blocked tasks. They extended the RTA by modelling the behaviour of the scheduler when it moves tasks from one list to another and the treatment of interrupts that preempts any task and the scheduler itself. Katcher et al. presented a framework based on the processor utilization factor with the same intention [44]. Both pieces of research constitute good examples of how to apply theoretical results in engineering.

Real-time system theory enables checking task sets that can meet all deadlines and allows simulating synthetic task sets for trace generation. For example, the system of Section 1.1 was simulated by the real-time simulator Cheddar. Having task sets capable of producing WSS sequences is required by MuSADET to detect timing anomalies. Since missed deadlines can be detected easily, we are not interested in including them under the MuSADET framework. After testing that simulated task sets can indeed produce WSS time-stamp sequences we moved onto working with traces generated by real-world systems.

The intuition that a real system with tasks having varying execution time per job instance will generate recurrent streams of events follows from the concept of the hyper-period. The RTS theory settles the foundation to determine task periods and compute other system properties from the trace. For example, in [40], we showed a method to mine task periods from traces. This method serves two purposes; it can be used to check for the system's specification or in the case of MuSADET to determine parameters for feature extraction.

2.2.2 Signal Processing

The purpose of using signal processing in MuSADET is the estimation of the features from timestamp sequences. We must consider that a scheduled task set is time-domain non-deterministic, and therefore their IATS should be considered realizations of a random process. Stationary or periodic IATS can be processed using power spectral density (PSD) estimation.

If an IATS originates in a WSS process, its first and second moments statistics will be on statistical equilibrium. That is, let $x_i[n]$ be an IATS generated by task τ_i, we say that $x_i[n]$ is WSS if; the mean and variance of $x_i[n]$ are constant, and the autocovariance of $x_i[n]$ is time-invariant.

$$\mathrm{E}[x[n]] = m_x, \text{ and } \operatorname{var}[x[n]] = \sigma_x^2 \tag{2.4}$$

Where $\mathrm{E}[x[n]]$ and $\operatorname{var}[x[n]]$ are the expected value and variance of the discrete sequence $x[n]$ respectively. The second condition needs the covariance of $x[n]$ and $x[m]$ to be a function of the lag between the two sequences $\ell \triangleq n - m, n > m$, that is:

$$c_{xx}[n,m] \triangleq \operatorname{cov}(x[n], x[m]) = c_{xx}[l] \text{ for all } n, m \tag{2.5}$$

Based on the concept of hyper period, Diaz et al. [27] showed that when the number of observed hyper periods tends to infinity, the distribution of response-times for a periodic task set becomes stationary. Response-times are an upper bound of time-domain task's activity. Therefore we can assume that IATS should also be stationary. The intuition comes from the fact that response-times are a subset of the events generated by a task. If response-times are stationary then the underlying generator must also be stationary.

The assumption that IATS are WSS enables the use of power spectral density estimation. Methods for PSD estimation can be classified into nonparametric and parametric [59]. The nonparametric approach does not assume there exists a model for the signal generator [64]. Since it is hard to establish a possible model for tasks as IATS generators, we focus on nonparametric PSD estimation. The methods under consideration are the periodogram [69] with its variations [4, 81], and the Blackman-Tukey method [7].

The power spectral density of a random process is the Discrete Time Fourier Transform (DTFT) of its autocorrelation sequence. It is a function of average signal power versus frequency (ω) [45, 59].

$$S_{xx}(\omega) \triangleq \sum_{l=-\infty}^{\infty} r_{xx}[\ell]e^{-j\omega\ell} \tag{2.6}$$

In MuSADET, we are interested in the PSD of IATS because the PSD could be used as the frequency domain signature of the generating task. The expectation is that the PSD will change if the behaviour of the task changes. The problem of (2.6) is that the analytic function of the autocorrelation is hard to determine. However, for stationary IATS, PSD estimation is possible from realizations of the process [59] and the periodogram.

$$\hat{P}(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \tag{2.7}$$

Equation (2.7) is the discrete-time version of the periodogram, which is a function of frequency. The periodogram was proposed by Schuster in 1898 [69]. Schuster used the periodogram to find *hidden periodicities* in astronomical and meteorological data sets.

Although a powerful tool, the periodogram is not a good estimator of the PSD. The problem is that the periodogram is not a consistent estimator of $S(\omega)$ [45]. We should notice that the periodogram is a biased estimator of $S(\omega)$ because $\mathrm{E}[\hat{P}(\omega)] \neq S(\omega)$. However, the periodogram is an asymptotically unbiased estimator of $S(\omega)$. That is, the expected value of the periodogram approaches $S(\omega)$ as $N \to \infty$ (where N is the length of $x[n]$) :

$$\lim_{N\to\infty} \mathrm{E}\left[\hat{P}(\omega)\right] = S\left(\omega\right) \tag{2.8}$$

The most problematic property of the periodogram is that its variance does not improve by increasing the length of $x[n]$. The variance of the periodogram as $N \to \infty$ is:

$$\lim_{N\to\infty} \mathrm{var}\left[\hat{P}(\omega)\right] = S^2\left(\omega\right) \tag{2.9}$$

A solution to the problem of PSD estimation by nonparametric methods consists of splitting the sequence $x[n]$ into a set of smaller segments. Then computing the periodogram of each segment and finally aggregating the results by averaging. Two classic methods that use this techniques are Bartlett [4, 3], and Welch [81, 82].

The Bartlett method of averaging periodograms divides the sequence $x[n]$ into a set of K segments of length M, where $x_k[m] = x[m+kM], k = 0, 1, \ldots, K-1, m = 0, 1, \ldots, M-$

1, then the periodogram is found for each segment using (2.7). Given that $\hat{P}_k$ is the periodogram of the k^{th} segment, the average of the periodograms is found by:

$$\hat{P}_B(\omega) = \frac{1}{K}\sum_{k=0}^{K-1}\hat{P}_k(\omega) \tag{2.10}$$

The Welch method divides $x[n]$ into a set of L segments of length M where segments are allowed to overlap, now each segment is formed as $x_k[m] = x[m+kD], k = 0, 1, \ldots, L-1, m = 0, 1, \ldots, M-1$, where in most cases $D \leq M$. For $D = M/2$ the overlap is 50%, being this the most common setting when using the Welch method. When $D = M$, there is no overlap and Welch is equivalent to Bartlett, that is, $L = K$. Instead of finding the individual periodograms using (2.7), Welch proposed the use of a modified periodogram by windowing the sequence $x_k[n]$, therefore:

$$\hat{P}_{mk}(\omega) = \frac{1}{MU}\left|\sum_{m=0}^{M-1} x[m]w[m]e^{-j\omega n}\right|^2 \tag{2.11}$$

Where U is the the normalization factor for the window $w[n]$:

$$U = \frac{1}{M}\sum_{m=0}^{M-1} w^2[m] \tag{2.12}$$

The Welch PSD estimate is found by averaging the modified peridograms (3.6):

$$\hat{P}_W(\omega) = \frac{1}{L}\sum_{k=0}^{K-1}\hat{P}_{mk}(\omega) \tag{2.13}$$

The last method to consider in this case is the Blackman and Tukey method proposed in 1958 (given reference is the corrected version published in 1959) [7]. The rationale of this method is to find the Fourier Transform of the windowed sample autocorrelation. Windowing the autocorrelation reduces the effect of large lags by giving a smaller weight to estimates for which ℓ is large. The effect of this windowing is a smoothing of the periodogram estimate, the formula is:

$$\hat{P}_{BT}(\omega) = \sum_{n=-(N-1)}^{N-1} r_{xx}[\ell]w[n]e^{-j\omega n} \tag{2.14}$$

Averaging or smoothing periodograms improves the consistency of the estimated PSD. The process comes with the drawback of trading improvement of the variance at the expense of frequency domain resolution. This issue becomes relevant when the signal under analysis contains components that are close in frequency because the method may fail to tell them apart. Increasing the length of the signal by padding will not improve the resolving capacity of any of these methods because there is no new information introduced to the signal.

Computational requirements are another factor to consider when using any of the methods shown in this thesis. MuSADET deals with a large number of signals with *concurrent* processing needs. Hence, we must take into account the total number of computations required to find PSD estimates. Table 2.1 includes a summary of the computational complexity for each of the methods presented above. Formulae in Table 2.1 use M radix-2 FFTs (i.e. M length is a power of two) computed over an N length sequence.

Table 2.1: Computational complexity of spectral estimation methods.

Method	Number of FFT	Complexity
Bartlett	$\approx \frac{N}{M}$	$\mathcal{O}\left(\frac{N}{2}\log_2 M\right)$
Welch 50% overlap	$\frac{2N}{M}$	$\mathcal{O}\left(N\log_2 M\right)$
Blackman-Tukey	$\approx \frac{N}{M}$	$\mathcal{O}\left(N\log_2 2M\right)$

This table is a compilation of computational complexities from Proakis and Manolakis, for details see: [64].

2.2.3 Distance Measures

In the previous section, we presented the fundamentals of the signal processing methods applied in MuSADET to extract FD features from IATS. We now present distances or similarities measures to compare power spectra. A distance measure or distance for short is a function that takes two vectors and outputs a scalar that measures how far apart are the input vectors. We formalize a distance function [26] as follows.

Let X, Y be two finite vectors, a measure or distance is a mapping function $D : X \times Y \mapsto [0, \infty)$. That is D takes in two real vectors and outputs a scalar representing the closeness or similarity between X and Y. A distance function should have some desirable properties to be of practical use:

- $D(X, Y) \geqslant 0$ **non-negativity**: This property, required by definition above and forces the distance function to be valid if and only if the formula maps to $[0, \infty)$ or if the input vectors meet the requirements to make the distance possible.

- $D(X, Y) = 0 \Leftrightarrow X = Y$ **Identity of indiscernibles**
- $D(X, Y) = D(Y, X)$ **symmetry**: Although a requirement for a distance to be **metric** it is not met by all distances.
- $D(X, Y) \leqslant D(X, Z) + D(Z, Y)$ **subaditivity or triangle inequality**: Another requirement for metric distances not met by all distances.

We present two classification categories for distances: (i) non-binary measures, for real-valued vectors; (ii) binary measures, for categorical data that takes on presence/absence (p/a) usually codified as $p = 1$, $a = 0$.

The term similarity is related to distance because many distances are equivalent to similarities through algebraic manipulation. For example, all distances that map to $[0, 1]$ become similarity by applying $S = 1 - D$. In the case of similarity, if the input vectors are equal then their similarity is $S_{x,y} = 1$. Due to the relationship between similarities and distances, many authors use the concepts interchangeably. We may also use the broader term **measure** to refer to distances or similarities as a general group of mapping functions.

Non-Binary Measures

MuSADET applies non-binary measures to PSD estimates (FD features). Frequency domain features in MuSADET are finite vectors, where each vector component is the power at a discrete frequency. When the distance measure is close to zero, the two PSDs are considered similar; if one of the PSDs is normal, then the other PSD should also be normal.

According to [14], non-binary measures can be catalogued in different families. The Minkowski family (2.15) [48, 26] generalizes some well known and useful distances, including the Euclidean. For example, $p = 1$ is known as Manhattan or City Block, $p = 2$ is Euclidean, and $p = \infty$ is the Chebyshev distance [14]. All distances in the Minkowski family are metric, i.e. they are symmetric and subadditive. In particular, Manhattan and Euclidean are the basis for other more elaborate distances to compare vectors of probability mass functions.

$$D_{Mink} = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p} \tag{2.15}$$

The intersection family [14] makes extensive use of Manhattan distance (Minkowski with $p = 1$) [29]. They find the similarity by considering the overlap (or lack of) of two vectors. For example, the non-overlap intersection (2.16) is more sensitive to elements of different sign and thus useful to detect changes in sign.

$$D_{non-Int} = \frac{1}{2} \sum_{i=1}^{d} |x_i - y_i| \tag{2.16}$$

The Wave-Hedges (2.17) [14] (also known as Soergel [26]) or the Czekanowski (2.18) distance applied to non-negative vectors, is bounded by the number of components. Each element in the sum is a normalized and independent, positive weighting coefficient. Therefore these two distances are appropriate when no component is considered dominant. Consequently, the choice of measure depends on the specific properties of the vectors under analysis.

$$D_{Hed} = \sum_{i=1}^{d} \frac{|x_i - y_i|}{\max(x_i, y_i)} \tag{2.17}$$

$$D_{Czek} = \frac{\displaystyle\sum_{i=1}^{d} |x_i - y_i|}{\displaystyle\sum_{i=1}^{d} (x_i + y_i)} \tag{2.18}$$

The χ^2 family [14] relies on the squared Euclidean distance (Minkowski with $p = 2$) [26]. The cornerstone of this family is Pearson χ^2 divergence, also known as Pearson distance (2.19), which is useful when comparing probability mass functions that follow $\chi^2(K)$ where K is the number of degrees of freedom.

$$D_P = \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{y_i} \tag{2.19}$$

The Pearson distance is non-symmetric and, therefore, in some cases, ill-suited as a feasible measure. Another related measure is Neyman's distance (2.20), which is also applicable in similar contexts to that of Pearson.

$$D_N = \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i} \tag{2.20}$$

Notice that $D_p(x, y) = D_N(y, x)$ and non-negative if $x_i, y_i \geqslant 0$ where $0 \leqslant i \leqslant N$, and N is the number of components of the two vectors. Both Pearson and Neyman distances meet the identity of indiscernibles property, therefore they can be combined to produce symmetric distances such as the max-symmetric χ^2 distance [14] shown in (3.14) and presented in Section 3.4.1.

Binary Measures

Measures for binary or categorical data can be represented using the Operational Taxonomic Units (OTU) notation [30] shown in Table 2.2. The notation has its root in taxonomic classification, where a significant number of these measures originated. Let X, Y be two finite vectors of attributes, a measure or distance compares X, and Y based on the selection of the properties that are relevant to the comparison. Therefore, selecting the appropriate distance or similarity is based on *making a sensible choice* [30].

One of the most common similarity distances for binary data is the Jaccard distance [30] that focuses on positive matches and the total number of mismatches (2.21). For example, let $X = \{1, 1, 0\}$, and $Y = \{1, 0, 0\}$, then $a = 1, b = 0, c = 1, d = 1$, and $D_{Jac} = \frac{1}{1+0+1} = \frac{1}{2}$.

Table 2.2: OTUs expression of binary instances.

	1 (presence)	0 (absence)	sum
1 (presence)	$a = \sum x_i y_i$	$b = \sum \neg x_i y_i$	$a + b$
0 (absence)	$c = \sum x_i \neg y_i$	$d = \sum \neg x_i \neg y_i$	$c + d$
sum	$a + c$	$b + d$	$n = a + b + c + d$

$$D_{Jac}(X, Y) = \frac{a}{a + b + c} \tag{2.21}$$

MuSADET employs a peak detection algorithm to find dominant peaks in normal PSDs, then transforms them into binary sequences. Therefore, we are interested in finding similarities between binary sequences. Since we are interested in matching dominant peaks, we favour positive matches while penalizing mismatches. Distances that take into account terms a and d in the numerator, while b and c in the denominator are of great interest.

Below, we include a selection of some binary distances. For a comprehensive list, please read [24].

$$D_{Sok}(X,Y) = \frac{a+d}{a+b+c+d} \text{ [74] (Sokal)} \tag{2.22}$$

$$D_{SokSne}(X,Y) = \frac{2(a+d)}{2a+b+c+2d} \text{ [74] (Sokal \& Sneath)} \tag{2.23}$$

$$D_{RogTan}(X,Y) = \frac{a+d}{a+2(b+c)+d} \text{ [76] (Rogers \& Tanimoto)} \tag{2.24}$$

2.2.4 Anomaly Detection

Many different strategies exist toward finding anomalies, most valid if applied to the appropriate type of anomaly and available data. In the ecosystem of anomaly detection methods, MuSADET is a classification engine based on time series. Our tool tries to classify timing anomalies in trace data converted to inter-arrival time series. Tools such as MuSADET train on a set of known labelled data to learn a model or classifier. After the classifier is learned it is used to label unknown data as normal or anomalous [18, 19].

The choice of approach for anomaly detection is highly dependent on the type of available data [19]. More than one technique applies to the same type of data, in the case of time series, some popular techniques are:

Similarity-based techniques treat the test sequence as a unit-element compared against the learned sequence. The classification depends on distance or similarity computed by some measure.

Window-based techniques compare a short window of symbols or short normal substring within the test sequence at a time. The window is usually shifted over the test trace to find anomalies.

Markovian techniques predict the probability of observing each symbol of the test sequence, using a probabilistic model, and use the per-symbol probabilities to obtain an anomaly score for the test sequence.

In MuSADET, we apply signal processing to extract FD features from IATS. After converting the IATS to frequency domain by a method such as Welch, we can rely on classification based on similarity/distance. Indeed, this choice of method is very convenient because the parameters to compute the power spectrum of the model and the test trace can simplify the classification process.

There are many measures useful for anomaly detection, recent surveys on this field report at least 76 for binary-based similarities [24] and 56 for the numeric (non-binary) class [15]. Similarity measures have been used to find sub-sequences or strings within a longer sequence [10]. A group of machine learning and clustering techniques using similarity distances are reported in two comprehensive surveys [19, 20].

Chapter 3

MuSADET-Framework

3.1 Overview

multi-signal anomaly detection for real-time traces (MuSADET)'s outline is shown in Figure 3.1. MuSADET analyzes event traces to detect timing anomalies. Traces can be produced within the system (e.g., operating system traces) or captured by an external trace logger (e.g., a controller area network (CAN) trace logger). MuSADET works under the semi-supervised paradigm [38], and thus, must be trained on traces collected during *well-behaved executions* of the system under analysis. There are four modules, as depicted in Figure 3.1, and two operational regimes, as explained below:

Training:

Before MuSADET can perform anomaly detection on test data, it must train a model of the system under analysis. The training stage requires a set of traces that are deemed *normal*. We assume that traces collected from a well-behaved system operating under normal conditions should mitigate the presence of anomalies.

Once the set of training traces is available, the training process can begin, with parsing being the first step. MuSADET will create a profile containing a description of the trace format and a data structure for the trained model. The analyst is responsible for providing the trace format of the system and the commands required to parse the training traces. During parsing, MuSADET will mine the set of generators and their signals. For each signal found in the trace, MuSADET will extract timestamp sequences and compute their inter-arrival times sequence (IATS).

The analyst can select signals for the system's profile, then MuSADET will compute and store the IATS of selected signals. Some IATS may not be suitable for feature extraction. For example, short signals containing a few time stamps for the whole trace can be excluded from the model. Short signals are usually the result of initialization or sporadic routines that do not repeat with a recurrent pattern and therefore do not produce recurrent streams of time stamps. The analyst can also exclude signals at will if they do not provide useful insight into the system's behaviour. Once the profiling is complete, the features extractor will compute the frequency domain features for each of the IATS and determine the appropriate method for classification. Finally, the trainer learns a model from the training features. The system's model is a collection of features associated with the set of signals and will be used to classify test traces when MuSADET operates in detection mode.

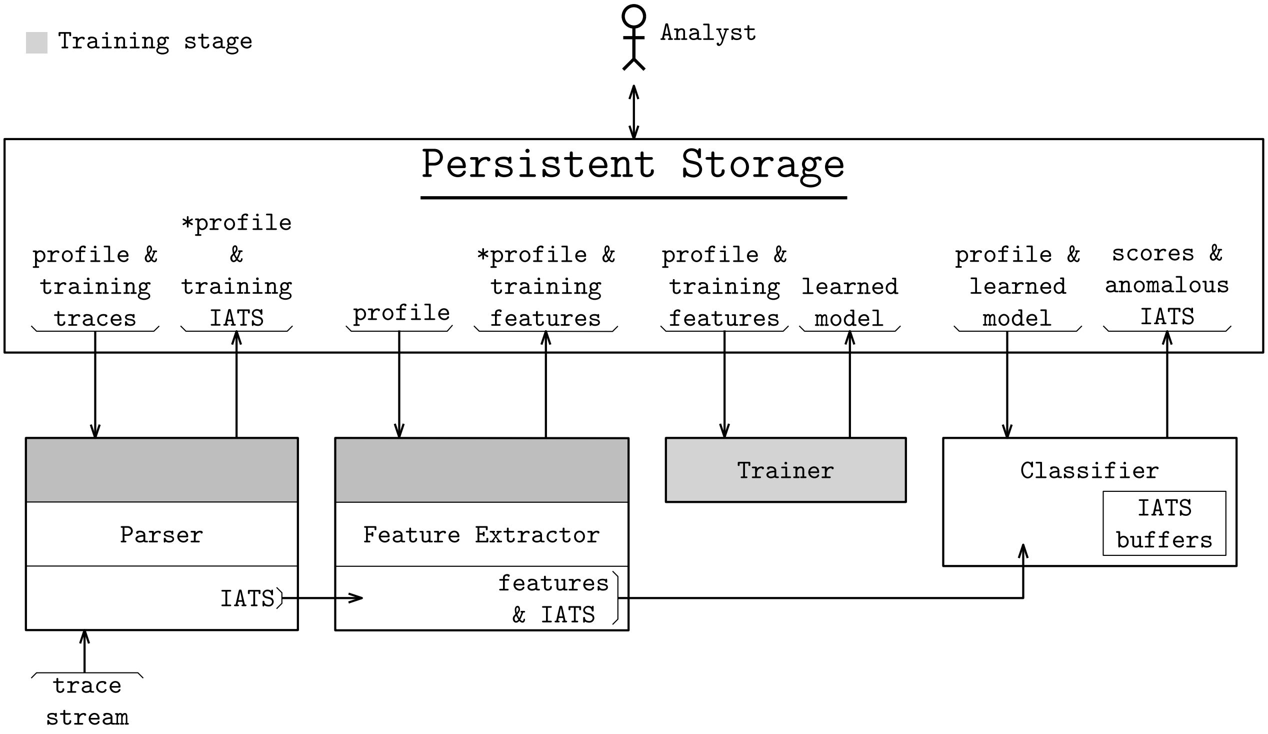


Figure 3.1: MuSADET's workflow.

Anomaly Detection:

In this mode, MuSADET parses the incoming stream of trace entries using the systems profile, which contains a description of the trace format and the set of signals to analyze. Since a trace can be of infinite length, MuSADET will process windows covering a fixed amount of time. The length of the detection window, stored in the profile, is a trade-off between feasibility and responsiveness. However, the detection window influences anomaly detection capabilities. For example, if the window is short compared to the average event rate, there would not be enough information to compute useful features. If the window is too long, anomalies can be washed out by long segments of normal data or the time to detection can be so large that the results may become useless.

After computing IATS from signals in the model set, MuSADET will determine if they meet the length and presence requirements. For example, if signals are missing in the test trace but are present in the profile of normal traces, they will be reported as an anomaly. For signals present in the normal model, MuSADET checks whether they are long enough to be processed by the features extractor. Signals failing the length requirement are also anomalous, but they do not requiere further analysis. Signals that appear in the test trace but are missing in the profile of normal traces are direct anomalies. We want to stress that anomalies are not indications of faults and, therefore, not necessarily linked to threats to the system. If, for example, MuSADET starts while the system boots, then signals related to the boot process will be missing, although the system may be operating normally.

The features extractor will process those IATS that meet the presence and length requirement. When in detection mode, the features extractor computes those features specified in the profile, this avoids wasting computing resources because not all features are suitable for the classification of all types of IATS. Once feature processing is complete, they are sent to the classifier. The classification process is feature dependent. For fully periodic IATS the method relies on the direct current (DC) significance method and the average amplitude of the IATS. If the IATS is non-DC then its classification will be based on its power spectrum and comparisons to equivalent features in the normal model.

3.2 Trace and Signal Models

Consider the trace snippet shown in Table 3.1. Each row or entry is a tuple composed of data points describing a single event that occurred at a particular time instant. For example, the trace entry e_1 records that the interrupt handler `0x44` preempted the process with $PID = 1$ at time $t = 2$. Events are activity markers of a generator's activity, and each generator can issue entries for several event types. Trace entries can share data values and thus, the parsing process must guarantee the extraction of consistent IATS. Of significant importance are those time entries having the same timestamp, like e_1 and e_2. For the case of e_1 and e_2, consistency is achieved by recognizing that the generators and events are different. We do not assume that timestamps will be unique because their value depends on the time resolution of the tracing tool, and therefore multiple entries may have the same timestamp.

Our goal is the extraction of IATS to model the behaviour of the system. The trace and signal models are as follows:

Definition 1 (Trace)**.** *A trace $\mathcal{T} \triangleq \langle H, B \rangle$ is a table with header H and body B. The header $H \triangleq \langle t, G, S \rangle$ of $\mathcal{T}$ is a tuple of attribute names where t is time stamp, G is the set of attribute names for generators, and S is the set of attribute names for signals. The body of $\mathcal{T}$ is $B \triangleq \{e_1, \dots, e_N\} : 1 \leq i \leq N$ a set of N trace entries.*

For example, the header of Table 3.1 is: $H = \langle t, \{\texttt{PID}, \texttt{TID}\}, \{\texttt{CLASS}, \texttt{EVENT}\} \rangle$, where $G = \{\texttt{PID}, \texttt{TID}\}$ and $S = \{\texttt{CLASS}, \texttt{EVENT}\}$. For this example, we need to combine the process identifier and the thread identifier as the generator identifier. The event identifier requires the combination of the event class and event type columns.

Definition 2 (Trace entry)**.** *A trace entry is a tuple $e_i \triangleq \langle v(t), v(G), v(S) \rangle$ of attribute values with attribute names $\{t, G, S\} \in H$.*

Definition 3 (Generator class)**.** *The generator class of $\mathcal{T}$ is $\mathbb{G} \triangleq \text{proj}(\mathcal{T}, G)$, where $\text{proj}(\mathcal{T}, G)$ is the projection of trace $\mathcal{T}$ on the set of attributes names $G \in H$.*

We range over the elements of $\mathbb{G}$ using the notation g_i where $g_i \in \mathbb{G}$ and $1 \leq i \leq |\mathbb{G}|$. The generator class contains all the identifiers for generators found within the trace. A generator is an independent entity that issues trace entries. For example, from the trace example of Table 3.1, we have processes and threads within processes. Since each process can spawn several threads we project both columns to obtain the generator class. Each generator $g_i \in \mathbb{G}$ can produce trace entries for a combination of several columns we identify as a signal, hence a generator has an associated signal class.

Definition 4 (Signal class)**.** *The signal class $\mathbb{S}_i$ of generator g_i is $\mathbb{S}_i \triangleq \text{proj}(\mathcal{T}, S, g_i)$, where $\text{proj}(\mathcal{T}, S, g_i)$ is the projection of trace $\mathcal{T}$ on the set of attributes names $S \in H$ given generator g_i.*

Table 3.1: Example of an event trace.

e_i	t	PID	TID	$CLASS$	$EVENT$
1	2	1		INT	0x44
2	2	1	1	THREAD	THREADY
3	3	1	1	THREAD	THRUNNING
4	12	1		INT	0x44
5	12	1	1	THREAD	THREADY
⋮	⋮	⋮	⋮	⋮	⋮
16	49	5		INT	0x49
17	49	5		COMM	SND_PULSE_EXE
18	49	5	1	THREAD	THRUNNING
19	49	1	1	THREAD	THREADY
20	49	5		COMM	REC_PULSE
⋮	⋮	⋮	⋮	⋮	⋮
32	54	1	1	THREAD	THRUNNING
33	54	1	1	THREAD	THREADY
34	56	1		COMM	SND_MESSAGE
35	56	1		COMM	REC_MESSAGE
36	58	1		COMM	SND_MESSAGE
⋮	⋮	⋮	⋮	⋮	⋮
39	59	1	4	THREAD	THREADY
40	59	1	4	THREAD	THRUNNING
41	59	1		COMM	REC_PULSE
42	60	1	4	THREAD	THRECEIVE
43	61	5		INT	0x49
⋮	⋮	⋮	⋮	⋮	⋮
72	72	1		INT	0x44
73	72	1	1	THREAD	THREADY
74	74	1	1	THREAD	THRUNNING
75	82	1		INT	0x44
76	82	1	1	THREAD	THREADY
⋮	⋮	⋮	⋮	⋮	⋮

The entry index numbers (e_i column) were added as a guide.

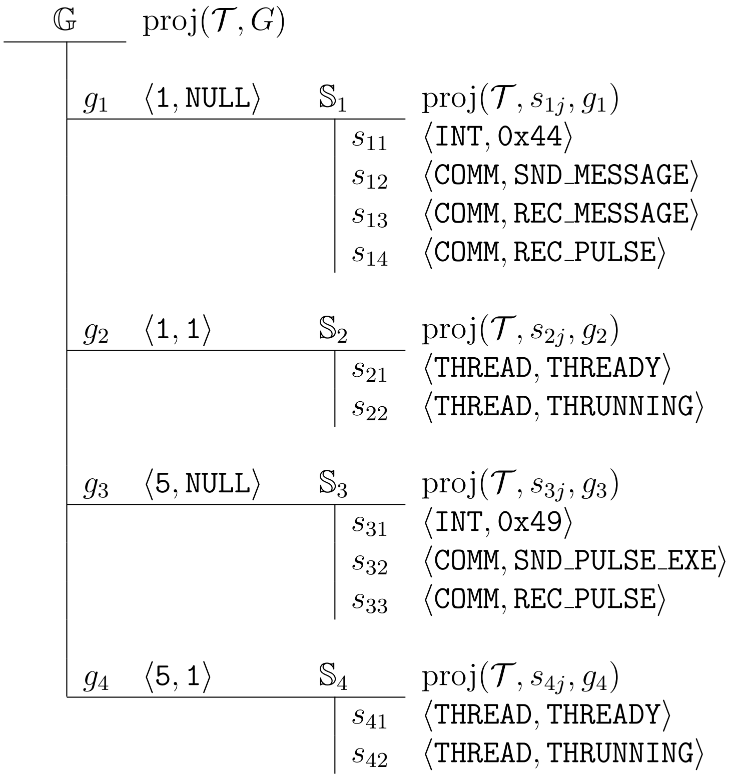

$\mathbb{G}$
$\text{proj}(\mathcal{T}, G)$
g_1 $\langle \texttt{1}, \texttt{NULL} \rangle$ $\mathbb{S}_1$ $\text{proj}(\mathcal{T}, s_{1j}, g_1)$
s_{11} $\langle \texttt{INT}, \texttt{0x44} \rangle$
s_{12} $\langle \texttt{COMM}, \texttt{SND\_MESSAGE} \rangle$
s_{13} $\langle \texttt{COMM}, \texttt{REC\_MESSAGE} \rangle$
s_{14} $\langle \texttt{COMM}, \texttt{REC\_PULSE} \rangle$
g_2 $\langle \texttt{1}, \texttt{1} \rangle$ $\mathbb{S}_2$ $\text{proj}(\mathcal{T}, s_{2j}, g_2)$
s_{21} $\langle \texttt{THREAD}, \texttt{THREADY} \rangle$
s_{22} $\langle \texttt{THREAD}, \texttt{THRUNNING} \rangle$
g_3 $\langle \texttt{5}, \texttt{NULL} \rangle$ $\mathbb{S}_3$ $\text{proj}(\mathcal{T}, s_{3j}, g_3)$
s_{31} $\langle \texttt{INT}, \texttt{0x49} \rangle$
s_{32} $\langle \texttt{COMM}, \texttt{SND\_PULSE\_EXE} \rangle$
s_{33} $\langle \texttt{COMM}, \texttt{REC\_PULSE} \rangle$
g_4 $\langle \texttt{5}, \texttt{1} \rangle$ $\mathbb{S}_4$ $\text{proj}(\mathcal{T}, s_{4j}, g_4)$
s_{41} $\langle \texttt{THREAD}, \texttt{THREADY} \rangle$
s_{42} $\langle \texttt{THREAD}, \texttt{THRUNNING} \rangle$

Figure 3.2: Example of a trace model.

We range over the elements of $\mathbb{S}_i$ using the notation s_{ij} where $s_{ij} \in \mathbb{S}_i$ and $1 \leq j \leq |\mathbb{S}_i|$. The signal class allows to differentiate among the different event types within a generator and enables the extraction of signals. Figure 3.2 shows the model for the trace snippet of Table 3.1. Notice that among the signal classes we can have equal elements e.g. $s_{22} = s_{42}$ but since messages of that type are generated by different generators we will obtain different signals.

Definition 5 (Signal)**.** Let $s_{ij} \in \mathbb{S}_i$ be a signal identifier for generator $g_i \in \mathbb{G}$. Signal $\mathcal{S}_{ij}$ is the subset of entries from $\mathcal{T}$ that match the given generator and signal identifiers. Formally: $\mathcal{S}_{ij} \triangleq \text{sel}(\mathcal{T}, g_i, s_j)$, where sel() is the selection function.

For example, from the model in Figure 3.2, given g_2 and s_{21} we obtain the signal $\mathcal{S}_{21} = \text{sel}(\mathcal{T}, \langle \texttt{1}, \texttt{1} \rangle, \langle \texttt{THREAD}, \texttt{THREADY} \rangle) = \{e_2, e_5 \ldots, e_{19}, \ldots, e_{33}, \ldots, e_{73}, e_{76} \ldots\}$ contains the trace entries generated by the generator with identifier $g_2 = \langle \texttt{1}, \texttt{1} \rangle$ and registered on the trace for the events with identifier $s_2 = \langle \texttt{THREAD}, \texttt{THREADY} \rangle$.

Definition 6 (Consistent signal)**.** Signal $\mathcal{S}_{ij}$ is consistent if and only if for every entry pair $e_i, e_j \in \mathcal{S}_{ij} : e_i.t \neq e_j.t$. That is, all entries in a signal must have different timestamps.

From Definitions 5 and 6, it is clear that choosing the set of attribute names for generators and signals requires careful analysis. For example, if $G = \{\texttt{PID}\}$ or $S = \{\texttt{THREAD}\}$ then the selection function of Definition 5 will return subsets with repeated timestamps. Such subsets of entries are not consistent signals because they violate a fundamental property of Poison renewal processes, which states that multiple instances of the same event cannot arrive at the same time [46].

Notice that from Definition 5, the timestamp (t) is the only attribute value that can change among any pair of entries taken from a well-formed signal. When $\mathcal{S}_{ij}$ is consistent, then the sequence $r_{ij}(\mathcal{S}_{ij}, t) \triangleq \mathrm{sort}(\mathrm{proj}(\mathcal{S}_{ij}, t))$, (i.e., the ordered projection of signal $\mathcal{S}_{ij}$ on the timestamp attribute name $t \in H$) can be modelled as a renewal Poisson point process.

Assumption 1 (TSS are WSS processes)**.** *Let r be the* real *time-stamp sequence (TSS) of signal $\mathcal{S}$ (for clarity, we dropped the sub-index ij). A renewal Poisson process for $\mathcal{S}$ can be defined in terms of r as: $R_n \triangleq \sum_{i=1}^{n} \mathbf{X}_i$, where $\mathbf{X}_1, \mathbf{X}_2, \ldots$ is the sequence of inter-arrival times, and each $\mathbf{X}_i$ is an independent identically distributed (IID)* random *variable [25, 46]. $\{\mathbf{X}_i\}$ is defined in terms of the arrival timestamps as $X_i \triangleq r[i] - r[i-1]$ for $i > 1$.*

For R_n to be stationary, the event's timestamps need to be independent. To meet this requirement, we do not allow the creation of signals that combine different events (see Definitions 4 through 5). For example, combining commands such as `file_open()`, `file_close()` will produce signals where timestamps are dependent on each other. Signal consistency (see Definition 6) is another requirement for stationarity. Because even though in theory, two events cannot occur at the same time, limitations of the tracing mechanism could generate timestamps of equal value. We also rely on the real-time system (RTS) theory principle of task's job independence stated in Section 2.2.1, i.e. consecutive jobs occur in the next period regardless of previous job execution.

Since R_n is stationary, it enables the use of power spectral density (PSD) estimation to compute the distribution of the signal's power over frequency from realizations of $\{\mathbf{X}_i\}$, which in turn allows the comparison of normal to test spectra. A realization $x[n]$ of $\{\mathbf{X}_i\}$ called IATS (IATS) can be extracted from finite instances ($r[m]$) of TSS.

Definition 7. The IATS $x[n]$ of TSS $r[m]$ is the N-length sequence after applying the mapping function $f[n] : r[m] \rightarrow x[n]$; where $f[n] \triangleq r[m] - r[m-1] \mid 1 \leq m < M$, $M = |r|$ is the length of $r[k]$, $N = M - 1$ is the length of $x[n]$, and $f[n]$ is the first difference function.

3.3 IATS Features

IATS belong to two classes:

Constant or DC sequences: This type of IATS results from specific design requirements. For example, instants of job release for periodic tasks in scheduled systems, sensor data transmitted on CAN-bus at a constant rate, periodic interrupts triggered by hardware timers.

Periodic and aperiodic sequences: Most systems will likely generate this type of IATS. Events can be deterministic to produce periodic IATS, or random to produce aperiodic IATS. These two types of IATS can be processed using PSD estimation.

3.3.1 Modelling IATS as a renewal processes

In Section we assumed that IATS are wide-sense stationary (WSS) sequences, in this section we show that IATS are indeed WSS sequences to which we can apply PSD estimation methods. We start analyzing periodic tasks under the worst case scenario and then extend the analysis to the general case. Our focus is on response times as they upper bound all other event generators for the same task. We also assume that the properties of the renewal process for response-times holds for any other event generated within the execution of a job. Extending the response-time analysis to other events requires that the average number of events generated within each job is constant over time.

We start our analysis by noting that a renewal process is an arrival process in which inter-arrival intervals are IID random variables. The rationale behind arrival theory is that the generating process *resets* at each arrival epoch. Renewal processes can be specified through the inter-arrival times $S_n = X_1 + \cdots + X_n$ or the underlying counting process $N(t), t \geqslant 0$. Consider the case where the nth arrival occurs at $t = \tau$, then counting k arrivals from $S_n = \tau$ we have $S_{n+k} - S_n = X_n + \cdots + X_{n+k}; X_i = S_i - S_{i-1}$. Hence, given $S_n = \tau, \{N(\tau + t) - N(\tau); t \geqslant 0\}$ we have a renewal counting process $\{N(t); t \geqslant 0\}$ in which $N(t)$ represents the number of arrivals to a system in the interval $(0, t]$ with $S_n \leqslant t$.

The counting process $N(t)$ and the inter-arrival process S_n are specifications that can be used to analyze the underlying renewal process. Each specification provides different possibilities, depending on the property of interest. In our case we want to determine if IATS generators are WSS. Notice that not all renewal processes meet stationary conditions. For example, aperiodic tasks such as error handlers or initialization routines can be modelled as a renewal process but in this case the expected value of the inter-arrival times is ill defined (i.e. the IATS has no mean). A renewal process S_n is WSS if it meets the following conditions:

- $\mathrm{E}\left[X_n\right] = \mu$ for all t. That is, the expected value of the inter-arrival times is constant.
- $\mathrm{K}_{XX}(t_1, t_2) = \mathrm{K}_{XX}(t_2 - t_1, 0) \triangleq \mathrm{K}_{XX}(t_2, t_1) = \mathrm{K}_{XX}(\tau)$. That is, the autocovariance function K_{XX} is time invariant.
- $\mathrm{E}\left[|X_n|^2\right] \leqslant \infty$. That is the expected value of the squared inter-arrival times is finite.

We start our analysis with the deterministic case where a task has periodic release time T and worst-case response time R. The random variable $X_n(t)$ for the inter-arrival time at epoch $n = N(t)$ is defined in terms of the inter-arrival process as $X_n(t) \triangleq S_n - S_{n-1}$. Considering R as the event of interest we have:

$$\begin{aligned} X_n(t) &= S_n - S_{n-1} \\ &= \left\lfloor \frac{t}{T} \right\rfloor T + R - \left(\left\lfloor \frac{t-T}{T} \right\rfloor T + R \right) \\ &= T \end{aligned}$$

where the nth release period is $N(t) = \left\lfloor \frac{t}{T} \right\rfloor$. The expected value of this process is:

$$\mathrm{E}\left[X(t)\right] = \mathrm{E}[\{X_n\}] = \frac{1}{n} \sum_{i=1}^{n} T_i = T$$

with variance $\sigma^2 = 0$. This result may seem trivial but if we allow $\sigma = c$ for small c we have a process where the expected value of inter-arrival times is still T with most its energy located at $\omega = 0$. Therefore, for a constant-rate generator there is no need to compute the PSD, it suffices to analyze the ratio of energy at DC to the total signal's energy. For a constant-rate generator, the proportion of energy that is accounted for the small variability of its inter-arrival times becomes noise in its power spectrum. More details about this particular type of inter-arrival process is presented in the following Subsection.

For the general case, we assume that the system total utilization is below one (see Equation 2.1) while the analysis of its worst-case response times meets the requirements stated in Section 2.2.1, Equation 2.3 or some other similar equation from the worst-case response family. In [28, 27, 57] Díaz, López et al. developed a stochastic framework for real-time systems and proved that the response-time distribution of schedulable systems is stationary. However, that does not mean the inter-arrival times are also stationary. From Figure 3.3 we see that a new job $\mathcal{J}_n$ is released with the task period. In the analysis of [27, 57] the response time $\mathcal{R}_n$ of the nth job depends on three factors: the pending workload at $t = nT$, the interference suffer by the job and the computation time required by the job. Using those three factors they proved that $\mathcal{R}_n$ is a random variable with stationary probability density function $F_{\mathcal{R}}$. In our analysis we do not need the usual index numbers for tasks and jobs.

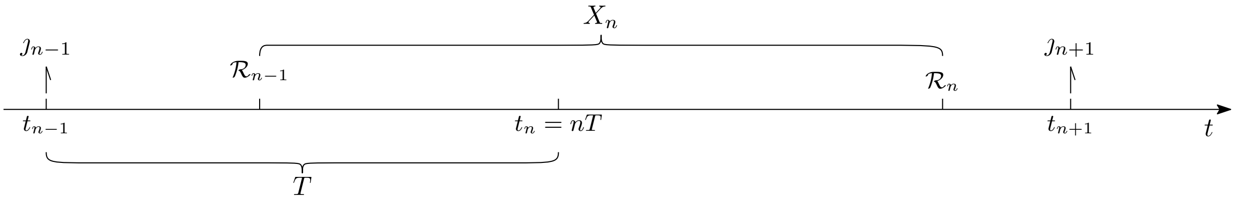


Figure 3.3: Response times and their inter-arrival time.

$$\begin{aligned}
X_n &= R(t_n) - R(t_{n-1}) \\
&= t_n + \mathcal{R}_n - (t_{n-1} + \mathcal{R}_{n-1}) \\
&= nT + \mathcal{R}_n - ((n-1)T + \mathcal{R}_{n-1}) \\
&= T + \mathcal{R}_n - \mathcal{R}_{n-1}
\end{aligned}$$

The expected value of X_n for sufficiently large n i.e. $n \to \infty$ can be computed as:

$$\begin{aligned}
\mathrm{E}[X_n] &= \frac{1}{n}\sum_{i=1}^{n} X_i \\
&= T + \frac{1}{n}\sum_{i=1}^{n}\mathcal{R}_i - \frac{1}{n}\sum_{i=1}^{n}\mathcal{R}_{i-1} \\
&= T + \frac{\mathcal{R}_n}{n} \\
&= T \qquad (3.1)
\end{aligned}$$

with initial condition $\mathcal{R}_0 = 0$.

There are two consequences of this result. First, the event expectation follows the job activation time. Second, if the event occurs multiple times within the execution of the job the expected value between the last occurrence of the previous job and first occurrence of the next job will dominate the average of the renewal process. That is, long renewals dominate over shorter ones. For theoretical explanation of this second consequence please read [70, 33]. The first consequence is evident in Figure 1.2, where the IATS mean is 15 time units regardless of the priority change. The second consequence is harder to show from IATS.

We now demonstrate that the autocovariance function is time invariant. Let $F_{\mathcal{R}}$ be the probability density function of response times for some task in the system. In [27, 57] Díaz, López et al. proved that $F_{\mathcal{R}}$ is stationary. From the autocovariance function definition we have $K_{XX}(t_1, t_2) \triangleq \mathrm{E}[(X_{t_1} - \mu_{t_1})(X_{t_2} - \mu_{t_2})] = \mathrm{E}[X_{t_1}X_{t_2}] - \mu_{t_1}\mu_{t_2}$. From Equation 3.1 we have that $\mu = T$ for all t, therefore:

$$\begin{aligned}
\mathrm{E}[(X_{t_1} - \mu_{t_1})(X_{t_2} - \mu_{t_2})] &= (X_n - T)(X_{n-1} - T) \\
&= [(T - \mathcal{R}_n) - T][(T - \mathcal{R}_{n-1}) - T] \\
&= \mathcal{R}_n\mathcal{R}_{n-1} \\
&= F_{\mathcal{R}}^2
\end{aligned}$$

also for all t. Note that the stationarity condition for $F_{\mathcal{R}}$ implies that the distribution of response-times is time invariant over a sufficiently long time interval. In fact the proofs in [27, 57] show that $\mathcal{R}_n$ becomes stationary over the task hyperperiod, consequently K_{XX} is also time invariant for IATS.

Assume we have a feasible system, let $X_n(t)$ be the inter-arrival time for the counting process $N(t)$ as shown before, then by definition $X_n(t) \leqslant \max[\{X\}]$ for all n and for any job in the system:

$$X_n(t) = T + \mathcal{R}_n - \mathcal{R}_{n-1} \leqslant T + R - C_0$$

where $\mathcal{R}_n$ is the response time of the nth job, T the task period, R the worst-case response time, and C_0 the best-case computation time of the task. Note that in this case the previous job had zero pending load and no interference, with minimum computational demand, i.e. $\mathcal{R}_{n-1} = C_0$. Neglecting the best case computation time, we maximize the inter-arrival time.

$$\max\left[\{X\}\right] \leqslant T + R - C_0 < T + R$$

by substituting R with the upper bound from [73, 6] we obtain:

$$\max[\{X\}] < \frac{T_i\left(1 - \sum\limits_{i<j} U_j\right) + \sum\limits_{i<j} C_j}{1 - \sum\limits_{i<j} U_j} < \frac{T_i + \sum\limits_{i<j} C_j}{1 - \sum\limits_{i<j} U_j}$$

where

$$R_i \leqslant \frac{\sum\limits_{i<j} C_j}{1 - \sum\limits_{i<j} U_j}$$

is the upper bound of response time, and U the utilization; the indexes i, j are included because the response-time bound at priority level i is calculated from the worst-case parameters of all tasks with priorities below or at level i. The utilization factor and the basic schedulability bound were introduced in Section 2.2.1.

Finally, we have that:

$$\mathrm{E}\left[|X_n|^2\right] < \mathrm{E}\left[|\max[\{X\}]|^2\right] < \infty$$

Since the three properties required for WSS stationarity have been demonstrated we can safely apply PSD estimation to IATS.

3.3.2 Ratio of DC to Total Power

An IATS generated by a constant-rate generator produces DC IATS. Some real systems like CAN-bus can generate events at an almost constant rate. A DC IATS concentrates all the signal power at $\omega = 0$. Therefore, we define the DC to total power ratio (DCTPR) as the ratio of energy at zero frequency to total signal energy. For DC IATS, an anomaly will be a drop in DCTPR for an IATS expected to be DC. We compute the DCTPR based on the energy preserving property of the Fourier transform:

Lemma 1. *DCTPR: The DC to total power ratio of IATS $x[n]$ is a measure of the amount of energy located at $\omega_k = 0$, it can computed as:*

$$D = \frac{\left|\sum_{n=0}^{N-1} x[n]\right|^2}{N \sum_{n=0}^{N-1} |x[n]|^2} \tag{3.2}$$

Proof. Let $x[n]$ be a finite length IATS with Discrete Fourier Transform (DFT) $X[k]$, its DC to total power ratio is defined as:

$$\begin{aligned} D &= \frac{|X[0]|^2}{\sum_{k=0}^{N-1} |X[k]|^2} \\ &= \frac{\left|\sum_{n=0}^{N-1} x[n]e^{-j0n}\right|^2}{\sum_{k=0}^{N-1} \left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2} \\ &= \frac{\left|\sum_{n=0}^{N-1} x[n]\right|^2}{\sum_{k=0}^{N-1} \left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2} \end{aligned} \tag{3.3}$$

From Plancheret-Parseval's theroem we have:

$$\begin{aligned} \sum_{n=0}^{N-1} |x[n]|^2 &= \frac{1}{N} \sum_{n=0}^{N-1} |X[k]|^2 \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \\ N \sum_{n=0}^{N-1} |x[n]|^2 &= \sum_{k=0}^{N-1} \left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \end{aligned} \tag{3.4}$$

Substituting the LHS of (3.4) in (3.3) we get:

$$D = \frac{\left|\sum_{n=0}^{N-1} x[n]\right|^2}{N \sum_{n=0}^{N-1} |x[n]|^2} \tag{3.5}$$

□

By using 3.2, we can compute the DCTPR from $x[n]$ in $\mathcal{O}(1)$ time complexity after the first N samples. When $D \approx 1$, the IATS has almost constant amplitude, and the DCTPR becomes the most reliable feature for anomaly detection. If an IATS expected to be DC under normal conditions becomes non-DC, its DCTPR will decrease because more energy distributes to other frequencies in the power spectrum. The DCTPR can detect deviations from DC by using the time-domain representation of $x[n]$. The DCTPR is superior to alternative methods such as comparing the incoming samples to the expected value and a set tolerance, thus removing tuning and human intervention.

3.3.3 Estimated Power Spectral Density

Consider the IATS shown in Figure 3.4; both originated on a stationary generator. The time-domain representations of the IATS in Figure 3.4 show noticeable differences while the power spectra are similar. From the similarities of the power spectra, we can conclude that despite the IATS differences, the generating process has maintained its temporal behaviour with respect to its generator.

Factors such as phase shifts in the generating process, small variations of inter-arrival times, and the first time stamp of the IATS leads to differences in IATS computed from the same generator. Issues such as those mentioned before do not affect the power spectrum of the IATS in frequency domain (FD). They mostly change the phase information of the signal in FD. Hence, we shift our attention to FD analysis because we can observe substantial differences in the IATS, and because the said differences do not necessarily correlate with timing anomalies.

When the IATS generating process is at least WSS, its PSD (i.e. power spectrum) has a defining FD *fingerprint* useful for anomaly detection. An anomaly will result in the loss or change of stationarity, thereby changing the PSD (i.e. the PSD changes its shape). Having different realizations of an IATS with different PSD shape is not enough for an unbiased comparison because IATS of different total energy are subject to scale differences and spectral leakage. Therefore, we seek a feature with the following basic properties:

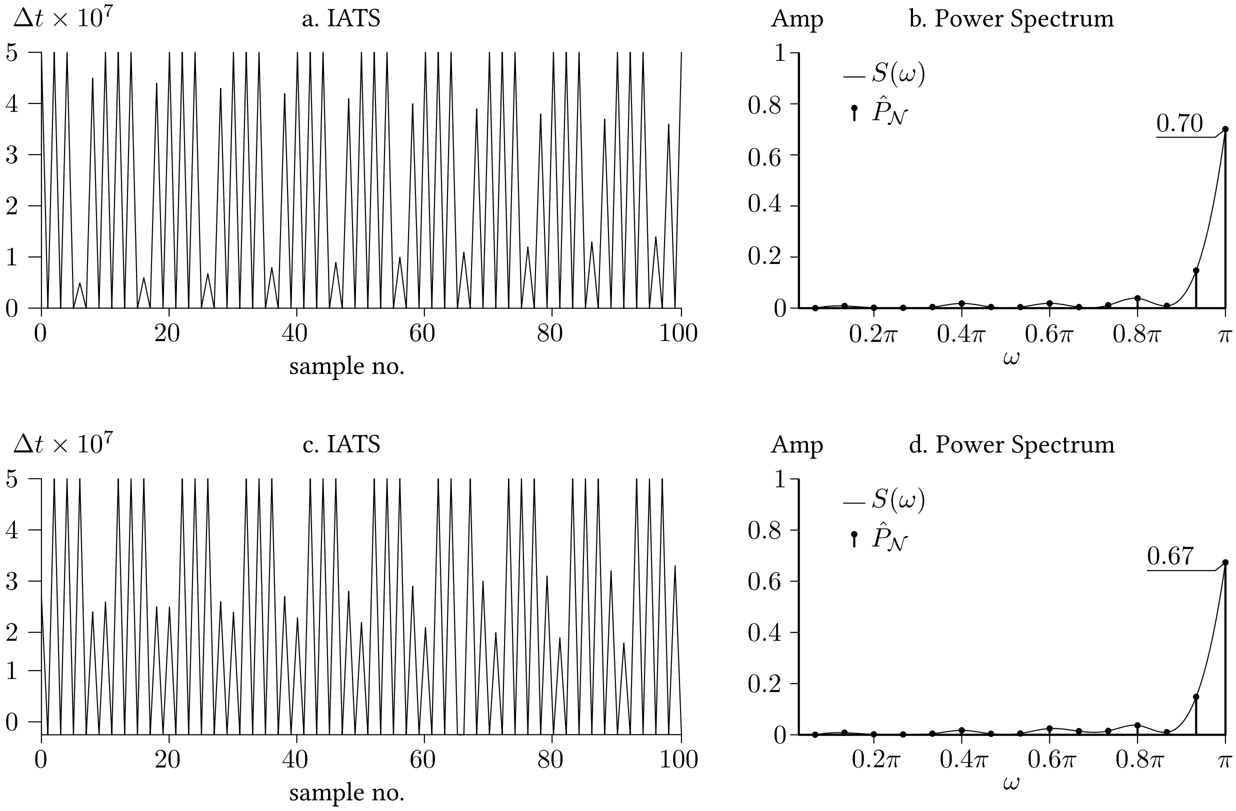


Figure 3.4: Example of IATSs and their power spectra.

Two segments of 100 samples of IATS from the same generator in a, and c with their corresponding normalized true PSD ($S(\omega)$), and Welch peridogram ($P[k]$) on the right. Note how although the IATS segments show differences, their power spectra are almost identical.

Scale independence

According to Parseval's theorem [61, 59], the Fourier Transform (FT) is energy preserving. Therefore the PSD of two sequences with similar frequency structure but different total energy are subject to scale differences. When computing model features, we use IATS computed from broader time windows than during anomaly detection. An IATS containing more samples increases the accuracy of the PSD estimate and reduces the effect of inconsistencies within the trace.

When MuSADET works in detection mode, we use narrower time windows for the IATS segments. The window length segment determines the time to detection, and thus, it should be as narrow as possible. This difference of window lengths used when MuSADET operates in different modes changes the total energy contained in the IATS of windows covering different time-spans. Hence, the method of choice must normalize the PSD amplitude while preserving the structure of power distribution in the frequency domain.

A Consistent estimate of the true PSD

The computed spectrum from the IATS must converge to the true PSD as $N \to \infty$, where N is the number of samples. This restriction serves two purposes: (i) by using longer sequences, more accurate PSD can be computed for the model, and (ii) it minimizes the mean square error (MSE) of the distance between similar PSD computed from sequences of different length.

Computing the PSD

In our case, we opted for the Welch periodogram because it estimates the PSD using overlapping segments of $x[n]$, which reduces the variance of the power spectrum. Another advantage of the Welch method is a reduction of spectral leakage due to windowing. The method to compute the Welch periodogram is as follows:

Let $x[n]$ be an IATS of length N. First, split $x[n]$ into a set of L overlapping segments of length M. Then compute the segment modified periodogram (we dropped the signal index to reduce notation cluttering):

$$\tilde{P}^{(l)}[\omega_k] = \frac{1}{MU} \left| \sum_{m=0}^{M-1} x^{(l)}[m] w[m] e^{-\mathrm{j}\frac{2\pi m}{M}} \right|^2, l = 0, 1, \ldots, L-1 \tag{3.6}$$

where $w[m]$ is a window to reduce spectral leakage and U is $w[m]$'s normalization factor. The Welch PSD estimate is found by averaging the modified periodograms computed using (3.6):

$$\hat{P}_{\mathrm{W}}[\omega_k] = \frac{1}{L} \sum_{l=0}^{L-1} \hat{P}^{(l)}[\omega_k] \tag{3.7}$$

More details on the Welch method can be found in [61, 59]. According to [13] scaling or normalizing the periodogram while preserving the FD structure can be achieved by using the sample variance of the times series ($\hat{\gamma}_0 = \text{var}[x[n]]$) as the normalizing factor:

$$P[\omega_k] = \frac{1}{L\hat{\gamma}_0} \sum_{l=0}^{L-1} \hat{P}^{(l)}[\omega_k] \tag{3.8}$$

The PSD estimate computed using (3.8) exhibits our sought properties. First, due to the averaging effect of the Welch method, $P[\omega_k]$ is a consistent estimate of the true PSD. Second, $P[\omega_k]$ is also scale-independent due to the normalization by $\hat{\gamma}_0$. We consider $P[\omega_k]$ as a defining feature for normal behaviour and use the notation $P[\omega_k]$, $P[k]$ or P, depending on or our need to specify the index of the frequency bin.

3.3.4 Binary Power Spectral Sequence

In [85], Mehdi et al. used the dominant peak of the FT for anomaly detection. They assumed that a recurrent generator would always have a fundamental frequency and, therefore a dominant peak in the amplitude of the FD representation of the IATS. We do not make such an assumption because the power spectrum of an IATS can contain more than one dominant peak. Instead, we analyze the PSD to determine a base threshold above which the PSD contains dominant peaks. We then transform the original PSD into a binary sequence where peaks above the threshold take the binary value one and the rest the value zero. We call this transformed sequence binary power spectral sequence (BPSS), which can then be compared to other BPSSs by a binary measure. Binary measures, when implemented as logic functions, reduce computational complexity. Also, the BPSS makes possible signal processing optimizations when computing the PSD of test IATS.

Let $\hat{P}_n[k]$, $\hat{P}_t[k]$ be the N-length PSD estimates of a normal, and test IATS respectively. Assume we want to determine how similar these two PSD estimates are. We need these two PSD in binary form to enable the comparison by a method like the Jaccard distance [24]. Therefore we can convert these PSD estimates to a binary sequence as follows:

Definition 8 (Binary power spectral sequence)**.** A binary sequence $B_P[n]$ such that it takes the binary value one if $\hat{P}[k] > Th_{AUTO}$ and zero otherwise.

$$B_P[n] = \begin{cases} 1, & \hat{P}[k] \geq Th_{AUTO} \\ 0, & otherwise \end{cases} \tag{3.9}$$

where Th_{AUTO} is a threshold above which the PSD has dominant peaks.

Classification using the BPSS focuses on the structural similarity of the power spectrum. That is, instead of checking the closeness of the signal's power at all frequencies, a binary measure, applied to the BPSS computes the distance based on the presence/absence of dominant FD peaks. For example, the plots in Figure 3.5 show the difference between PSD and BPSS. Notice that in the BPSS, all frequency components bellow the threshold are treated as noise.

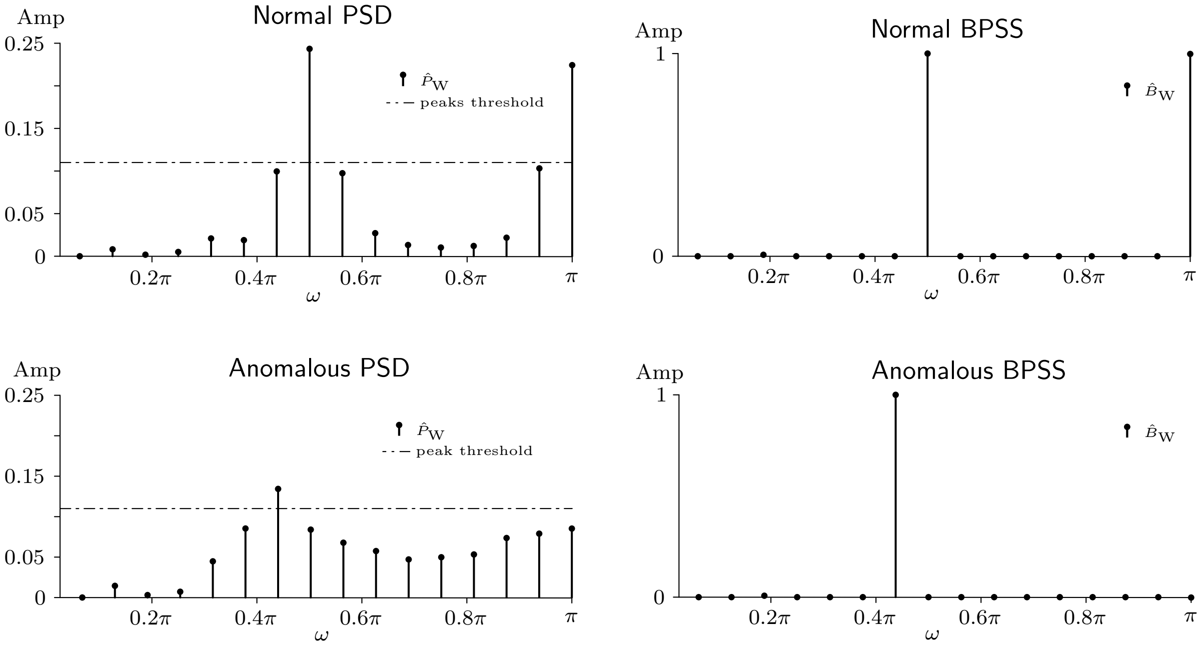


Figure 3.5: Binary Power Spectral Sequence.

A normal and an anomalous PSD on the left with their corresponding BPSS on the right. Note the missing dominant peak in the bottom right plot of the anomalous BPSS.

Peak detection

To detect dominant peaks, we apply `BPSS_THRESHOLD` (Algorithm 1), an adaptation of the `PERIOD_HINTS` algorithm. Proposed by Vlachos et al. [80], `PERIOD_HINTS` determines a threshold above which any component in the power spectrum becomes a period hint. A period hint is a dominant FD peak that may be associated with a real periodicity of the stationary signal. PERIOD_HINTS uses the raw periodogram and contains other features not needed in MuSADET. For example, PERIOD_HINTS may discard some hints if they are likely to represent unreal periodicities. In MuSADET we cannot assume that there are underlying periodicities such as seasonal cycles and therefore we include all dominant peaks in the model.

MuSADET must handle the presence of low-frequency components that do not provide useful information regarding the true periodicities of the generator. That is, there may be dominant peaks in the low-frequency range due to very long periodicities or leakage. We approach this issue by filtering the time series and applying the Welch method to compute the PSD. The use of Welch periodogram reduces leakage while low-frequency peaks are removed by filtering the IATS with a high pass filter. BPSS_THRESHOLD returns t_h the threshold above which FD peaks become dominant and the BPSS of IATS x.

Algorithm 1 BPSS_THRESHOLD

```
Require: $x, h, M$
Ensure: $t_h$, $b$ {Threshold of significant peaks and BPSS }
$P_{max} = \emptyset$ {Vector of maximum peaks}
$x_f = \texttt{filter}(x, h)$

for $l = 1$ to $100$ do
    $x_l = \texttt{permute}(x_f)$
    $\hat{W}_l = \texttt{welch}(x_l, M)$ {segment length: $M$, overlap: 50%}
    $\texttt{insert}(P_{max}, \texttt{max}(\hat{W}_l))$
end for

$\texttt{sort}(P_{max}, \texttt{descending})$
$t_h = P_{max}[5]$ {$5^{th}$ element is $95^{th}$ largest power}

$\hat{W}_W = \texttt{welch}(x_f, M)$

for $k = 1$ to $\left|\hat{W}_W\right|$ do
    $b[k] = (\hat{W}_W[k] \geqslant t_h)\ ?\ 1 : 0$
end for

return $b, t_h$
```

To detect dominant peaks, BPSS_THRESHOLD takes three input parameters: (i) an IATS x, (ii) a high pass filter kernel h, and (iii) the segment length for the Welch periodogram. First, the algorithm filters x to obtain x_f (line 2). Then it randomizes the content of x_f to produce x_l. The scrambling process should destroy the frequency domain structure of x_f, and therefore the PSD of x_l would not resemble the PSD of x_f. The maximum power of $\hat{W}_l$ is then stored in the power array P_{max}. The scrambling of x_f, PSD estimation, and maximum power picked (lines five through seven) is repeated 100 times to produce periodograms without the FD structure of x_f. Finally, the algorithm will set the threshold to the fifth higher power in P_{max}, which should be enough to separate dominant and non-dominant peaks in the PSD of x_f.

Computational advantage of BPSS over PSD estimates

By using BPSS instead of PSD we can reduce the computational complexity of MuSADET's anomaly detection on non DC signals. However, when using BPSS we sacrifice feature accuracy for anomaly detection performance. The differences in anomaly detection performance for our case study are presented in Section 4.3. The improvements on computational complexity we discuss below only apply when MuSADET works on detection mode. During the training stage the model BPSS must be computed as described before and therefore training on BPSS features requires more computations than PSD.

We start our discussion by noticing that the BPSS feature would contain a few dominant peaks. For example, the normal BPSS in Figure 3.5 has two dominant peaks out of sixteen frequency components. The BPSS feature heavily relies on the Fast Fourier Transform (FFT) to compute the power spectrum for each of the windows required to compute the Welch periodogram. Training provides model BPSS from which we can extract which frequency components should have dominant peaks. Computing a few DFT points can reduce the number of computations required to extract the feature from analysis IATS. Hence, we require an algorithm that can compute K frequency components of the power spectrum with a lower computational load than $\mathcal{O}(M \log_2 M)$. Where K is the number of frequency components and M the length of the sequence.

We shall note that direct computation of the M-point DFT is in $\mathcal{O}(M^2)$, and the FFT computes all DFT coefficients in $\mathcal{O}(M \log_2 M)$. The Goertzel [37] algorithm can compute a few samples of the DFT with more efficiency than the FFT but will require $\mathcal{O}(M^2)$ if all DFT points are computed with this method. The FFT works as a *block* algorithm and its computing cost is fixed for a given sequence length.

Let K be the number of dominant peaks in a model BPSS the Goertzel algorithm is more efficient than the FFT when $K < \log_2 M$ [59]. We start the analysis by noting that:

$$e^{j\frac{2\pi kM}{M}} = 1$$

Then the DFT of $x[n]$ can be written as:

$$\begin{aligned} X[k] &= e^{j\frac{2\pi kM}{M}} \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi kn}{M}} \\ &= \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi}{M}k(n-M)} \\ &= \sum_{n=0}^{M-1} x[n] e^{j\frac{2\pi}{M}k(M-n)} \qquad (3.10) \end{aligned}$$

Equation 3.10 can be transformed into a recursive filter as described in [79, 59] where:

$$\begin{aligned} y_k[n] &= e^{j\frac{2\pi kn}{M}} y_k[n-1] + x[n],\ 0 \leqslant n \leqslant M \\ X[k] &= y_k[M] \end{aligned}$$

with initial conditions $y_k[-1] = 0$, and input $x[n] = 0$ for $n < 0$ and $n \geqslant M$. The Goertzel algorithm expressed in recursive form requires M complex multiplications to compute one DFT value. Hence if K DFT points are needed computational complexity is $\mathcal{O}(KM)$ and Goertzel outperforms the FFT when $K < \log_2 M$. A further improvement, although not significant in terms of algorithmic analysis, could take advantage of DFT points computed for previous windows at the expense of storage cost. Since the Welch periodogram splits a window of length N into smaller overlapping windows. Storing DFT values of previously computed significant peaks can improve performance and enable the use of a sliding window with reduced cost when averaging a few frequency components.

In this thesis we did not compute the BPSS using the Goertzel algorithm. By applying the Goertzel method we would assume that only normal dominant peaks will be above the peak threshold. Although normal IATS should only have peaks at its dominant frequency components it is not the general case for all IATS, making Goertzel only valid to classify normal IATS. Moreover, to use the Goertzel algorithm the violation of its assumption must be taken into consideration. For example, by checking if there are chances to have other dominant peaks at frequency components that should be below the threshold. Validating the said assumption could be done by applying Parseval's theorem after computing each PSD point to determine if there is more energy left in the power spectrum such that a peak cannot be above the threshold. Notice that applying Goertzel would require a strategy to either include more frequency components until no more peaks can go beyond the threshold or switch to the PSD method when the assumption cannot be satisfied with absolute certainty. Hence, we limited our work to verify whether or not the BPSS is a feasible feature for anomaly detection by first computing the PSD and then transforming it into the BPSS.

3.4 Classification of test features

We detect anomalies present on test IATS by comparing its defining feature to a model. For non-DC IATS $\mathcal{M} \triangleq \{P_1, \ldots, P_N\}$ is a set of N PSD estimates. We present the classification of non-DC signals first.

3.4.1 Classification by χ^2 distances on PSD features

Consider a feature $P[k]$ computed from Equation (3.8), and the random variable $\mathbf{P}_k$, the signal power at ω_k. Since $x[n]$, the IATS generated by a recurrent generator is a WSS stationary sequence, we can assume that $\mathbf{P}_k$ is normally distributed with sample mean $\overline{\mathbf{P}}_k = \frac{1}{L\hat{\gamma}_0} \sum_{l=0}^{L-1} \hat{P}_{ij}^{(l)}[\omega_k]$ and sample variance $s_k^2 = \text{var}[{}^{\hat{P}_{ij}^{(l)}[\omega_k]}/_{\hat{\gamma}_0}]$, hence:

$$\mathbf{P}_k \sim N(S(\omega_k), \sigma^2(\omega_k)) \tag{3.11}$$

where $S(\omega_k)$ is the true mean power with variance $\sigma^2(\omega_k)$ at frequency $\omega = \omega_k$. Since $\mathbf{P}_k$ is normally distributed, then $\mathbf{P}_k^2 \sim \chi^2(1)$. Let $\mathbf{W}_K = \mathbf{P}_1^2 + \mathbf{P}_2^2 + \cdots + \mathbf{P}_K^2$, it is known that $\mathbf{W}_K \sim \chi^2(K)$ where the χ^2 metric can be computed by:

$$Q_P = \sum_{k=1}^{K} \frac{(\mathbf{P}_k - S(\omega_k))^2}{S(\omega_k)} \tag{3.12}$$

Equation 3.12 is also known as the Pearson's-χ^2 distance and can be used to measure the distance between features P_a, and P_i as:

$$Q_P(a,i) = \sum_{k=1}^{K} \frac{(P_a[k] - P_i[k])^2}{P_i[k]} \tag{3.13}$$

where P_a is the analysis feature and P_i is the model feature. The $Q_P(a,i)$ statistic in (3.13) can be approximated by a $\chi^2(K-1)$ distribution and is a measure of the discrepancy between the observed feature P_a and the expected feature $P_i \in \mathcal{M}$.

Since (3.13) is a nonsymmetric distance, it can produce false results even when the compared spectra are dissimilar. To avoid this problem we take the max-symmetric χ^2 distance [14] defined as the maximum between the χ^2-Pearson's and χ^2-Neyman's distances:

$$V(a,i) = \max\left(Q_P(a,i), Q_N(a,i)\right) \tag{3.14}$$

where the χ^2-Neyman distance $Q_N(a,i)$ is:

$$Q_N(a,i) = \sum_{k=1}^{K} \frac{(P_a[k] - P_i[k])^2}{P_a[k]} \tag{3.15}$$

Computing classification score

Note that applying the method described above produces $|\mathcal{M}|$, χ^2 distances. Hence after choosing distance $V(a,i) : 1 \leqslant i \leqslant |\mathcal{M}|$ we can use a p-value and the $\chi^2(K-1)$ distribution to assign the final classification.

If $V(a,i) \leq \chi^2_p(K-1)$ there is no evidence to reject the null hypothesis that the features are equivalent, whereas if $V(a,i) > \chi^2_p(K-1)$, it is unlikely that the differences between the features are due to random chance. Formally:

$$\mathcal{H}_0 : \quad V(a,i) \leq \chi^2_p(K-1)$$

$$\mathcal{H}_a : \quad V(a,i) > \chi^2_p(K-1)$$

Based on the test above we assign a classification score as:

$$v_i = \begin{cases} 1 & V(a,i) > \chi^2_p(K-1)(anomalous) \\ 0 & otherwise(normal) \end{cases}$$

Selecting $V(a,i)$ for classification presents a variety of options. The appropriate index $1 \leqslant i \leqslant |\mathcal{M}|$ of the model feature taken as a reference to assign the score would depend on the sensitivity level or decided based on efficiency needs for an online anomaly detection scenario. For example, selecting $max(V(a,i))$ would be pessimistic given the model and comparison feature. Another approach could classify the feature as anomalous as soon as some $V(a,i) > \chi^2_p(K-1)$ meaning that we accept the feature to be normal if an only if all distances are smaller than $\chi^2_p(K-1)$, which is also a pessimistic approach. Selecting the median distance $\tilde{V}(a,i)$ is neither pessimistic nor optimistic. When using the median distance method an odd number of model features ensures that a χ^2 value is selected. The last two methods help to reduce the number of distances required to determine the score. We used the middle-point distance method in our evaluation of MuSADET.

3.4.2 Classification by Jaccard distance on BPSS features

Consider a feature B_P computed by Equation (3.9), it takes the value one for significant peaks of the PSD, and zero otherwise. The BPSS can now be compared to model features by a binary measure. The choice of measure will determine the effectiveness of the similarity test. A normal BPSS should contain a small number of dominant peaks. That is, most frequency components of model features would be zero. The bias of the BPSS toward significant peaks determines the choice of measure.

In Section 2.2.3, we presented some widely used binary measures. We also stressed the relevance of matches and mismatches. After testing some binary measures, we concluded that the best choice would be the Jaccard similarity. The rationale for our choice stands on how the Jaccard similarity operates. Positive matches of the BPSS, (i.e. matching dominant peaks) are divided by the total of matches and mismatches. The number of matches in the denominator forces the similarity to be one when both features are the same. Even if the features have the same dominant peaks, any mismatch will reduce the similarity. For example, consider the BPSSs as shown in Figure 3.5 they share one dominant peak, and there is a mismatch at the last frequency component, therefore $D_{Jac}(a,i) = \frac{1}{2}$, which in this case means that both sequences are dissimilar.

Computing classification score

To compute the score we apply (3.16) to all model BPSSs comparing them with the test BPSS

$$D_{Jac}[i](B_{Pa}, B_{Pi}) = \frac{a}{a+b+c} \tag{3.16}$$

to obtain a set of similarities, where B_{Pa} is the analysis feature, $B_{Pi} \in \mathcal{M}$ is the model feature, and a, b, c are computed according to Table 2.2.

Note that applying the method described above produces $|\mathcal{M}|$, Jaccard similarities. To compute a score from this set of similarities, we apply the harmonic mean, which is appropriate to compute the average of rates. Notice that the Jaccard similarity can be seen as a rate. In our case, the harmonic mean is less sensitive when the similarities are close to one but reacts strongly when any similarity is close to zero. The score based on BPSS is computed as:

$$s_B = \frac{N}{\sum_{i=1}^{N} \frac{1}{D_{Jac}[i]}} \tag{3.17}$$

Note that if any of the similarities is zero then its reciprocal in Equation (3.17) would be undefined, therefore in case a similarity is zero we assign a real number close to zero to overcome division by zero.

3.4.3 Classification by DCTPR

Despite the power of DCTPR to detect departure from DC, it is not enough to classify the signal. For example, a misbehaving generator that doubles the rate of an event will have the same DCTPR, but the average of the IATS will be half of the expected value. Therefore, in the presence of a signal with normal DCTPR we also check that the mean value of the IATS amplitude is within a specific range. Hence, we assign an anomaly score score for DC signals as follows:

$$S_D = \begin{cases} 0 & D_a \geqslant T_h \wedge \bar{x}_a \in [\bar{x}_a \pm n s_M]\,(normal) \\ 1 & otherwise(anomalous) \end{cases}$$

where $T_h : T_h < 1$ is the classification threshold for the DCTPR, s_M is the standard deviation of the IATS in the model and $n > 0$ is a scaling constant.

The threshold T_h sets how much energy needs to be concentrated at $\omega = 0$. If T_h is too low, it would consider signals with significant non-DC peaks as DC while a too high threshold would force classification by the χ^2 metric of signals whose non-DC part of the power spectrum is composed of noise.

We do not propose any automated method to tune T_h but we found that $T_h = 0.9$ (i.e., at least 90 % of the signal's energy must be DC) as threshold worked well for the datasets we present in our case studies. The term ns_M sets the tolerance variation of the mean value of the IATS. For DC signals s_M is always relatively small, and we found that for $n = 1$ classification results were accurate.

Chapter 4

Case Studies

In this chapter, we present and discuss the results of applying multi-signal anomaly detection for real-time traces (MuSADET) to detect timing anomalies present in datasets from two different systems. The HCRL CAN[1] injection dataset contains traces collected from the controller area network (CAN) deployed in a vehicle. The HEXACOPTER[2] dataset is composed of traces collected from a QNX[3]-controlled hexacopter. Both datasets contain normal data from which we trained normal models. After training the model, we performed anomaly detection on test data for which we present two analysis techniques. The performance analysis evaluates the quality of anomaly detection while the visualization technique shows how our tool can be useful in an engineering environment. We also provide a comparison between MuSADET and the related method Signal Processing for Trace Analysis (SiPTA), where we show that MuSADET outperforms SiPTA. The following section presents a brief introduction to performance analysis and our choice of tools or indicators. We also introduce a graphical method to present anomaly scores to an analyst. Such visualizations are useful because they can show how some generators and signals are affected by present anomalies.

[1] An in depth description of the HCRL CAN dataset can be found at: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset

[2] Contact information for inquiries about the HEXACOPTER dataset can be found at: https://uwaterloo.ca/embedded-software-group/

[3] http://blackberry.qnx.com/

4.1 Performance Analysis

We provide performance analysis on anomaly detection for both case studies. Since MuSADET is a binary classifier, the results of a particular classification setup depends on the threshold that divides the data into normal/anomalous. For each possible threshold value, there is a particular distribution of predictions. Rates are computed from the confusion table, as shown in Figure 4.1.

The first analysis method we show is the receiver operating characteristic curve (ROC) [9, 31], which provides an intuitive graphical representation for threshold levels that cover the scores range. The second analysis takes the best threshold value and analyzes the performance using indicators (measures) that assess the quality of the classifier from different viewpoints. The following subsections provide a brief introduction to the ROC curve and the indicators we use.

		True class	
		positive: P	negative: N
Predicted class	PT	True positive TP	False positive FP (Type I error)
	PN	False negative FN (Type II error)	True negative TN

Rates

True positive rate

$$TPR = \frac{TP}{P}$$

False positive rate

$$FPR = \frac{FP}{N}$$

False negative rate

$$TNR = \frac{FN}{P}$$

True negative rate

$$TNR = \frac{TN}{N}$$

Figure 4.1: Confusion table and rate equations.

Confusion table on the left and the corresponding rate equations on the right.

Receiver Operating Characteristic

The ROC curve [9, 31] is a well-known method to analyze the performance of classifiers. The ROC curve is attractive due to some desirable properties. (i) it is a compact and normalized graphic representation of classification rates, (ii) it is insensitive to change in the class distribution, (iii) it is suited for discrete and continuous classifiers. ROC curves provide an intuitive representation for classification performance as detection threshold changes. In this thesis, we use the ROC to show how the classifiers under analysis (MuSADET or SiPTA) performs when detecting anomalies from different generators and signals.

The ROC curve is built from the anomaly detection results obtained after applying MuSADET or SiPTA to windows (segments) of a trace. Recorded results for each window contain computed scores for signals and the window's true value. Based on the threshold level, each window receives a class assignment. The first threshold being the minimum score among all windows. The results are cast into the confusion table, and the computed FPR and TPR become the coordinates of the ROC for that particular threshold. The threshold is then increased by a set amount and the computations involving the confusion table repeats to cover the range of scores.

The process to construct a ROC curve yields a graph such as the one shown in Figure 4.3. For this example, the threshold ranges over the interval $Th \in [0, 1]$ where $Th = 1$ means that a window of a particular signal will be classified normal if the DC to total power ratio (DCTPR) score is exactly one and anomalous otherwise. The point for $Th = 1$, located at coordinate $(0, 0)$, means that all windows are considered normal, therefore $TPR = 0$ and $FPR = 0$. That is, no anomalous windows are found, and no normal windows are misclassified. Likewise, the point for $Th = 0$ located at coordinates $(1, 1)$ means that all windows are considered anomalous, therefore $TPR = 1$ and $FPR = 1$. That is, all anomalous windows are found, but also all normal windows are misclassified. The perfect classifier would have a point at coordinates $(0, 1)$ where all true positives are found without any false positive. Desirable points are those near coordinate $(0, 1)$, where the true positive rate is maximized while the false positive rate is minimized.

Performance measures

Several measures provide insight into how certain classifier performs. The results computed from a confusion table are the first group of such measures. The problem is that they alone do not provide a complete picture of the classifier. Performance measures such as accuracy, precision, recall, and F1-scores, complement the ROC curve. These metrics enable comparisons between SiPTA and MuSADET. They also allow comparisons within MuSADET. Since MuSADET is a multi-signal classifier, we are also interested in its performance for different signals. We report performance measures in a table such as Table 4.3 we also included a brief description of some measures.

Precision: $Pr = \frac{TP}{TP+FP}$ quantifies the quality of positive predictions.

Recall: $Rc = \frac{TP}{P}$ quantifies the retrieving power of the classifier.

Accuracy: $Acc = \frac{TP+TN}{P+N}$ is an overall measure that quantifies the capacity of the detector to properly identify both positive and negative cases from the overall population.

F1-score: Defined as $F1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$ this measure combines the two indicators aimed at positive classification using the harmonic mean. The F1-score is known to be biased toward positive classification.

Mathews Correlation Coefficient: $Mcc = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$ combines all the elements of the confusion table and therefore includes both negative and positive predictions as a measure of classification quality. The MCC is also a balanced measure and works quite well even when the classes are of different size.

4.2 HCRL CAN injection

The traces in this dataset were collected by the Hacking and Countermeasure Research Lab (HCRL-South Korea). Table 4.1 is the summary of the HCRL dataset. Each trace contains two segments. The first segment includes normal and anomalous messages injected into CAN. Following the anomalous fragment, there is a segment of normal CAN data. The conditions in which the vehicle operated during the experiments are unknown. So we presume that for safety reasons the vehicle was stationary during the collection of anomalous data.

4.2.1 Brief introduction to CAN-bus

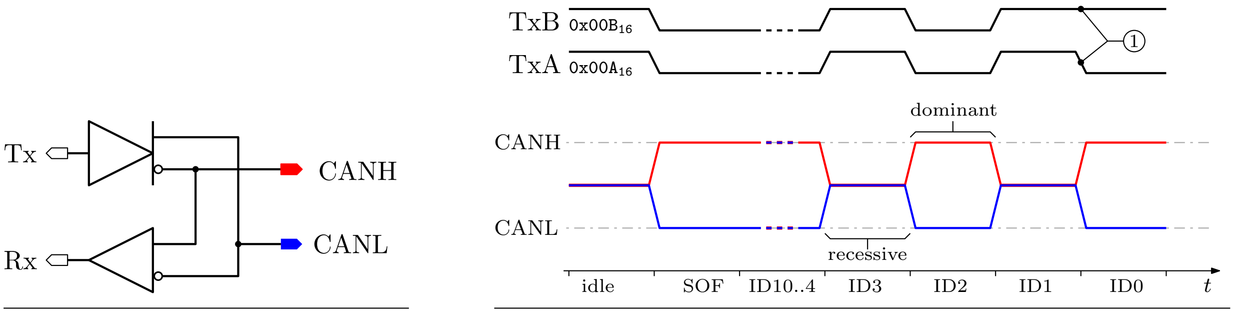
The CAN communication protocol (ISO 11898) is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD/AMP) [63, 75]. Commonly known as CAN bus it is mostly used in the car industry for which it was originally designed. CAN bus is fast, robust and low cost due to its simplicity and the need of only two wires for communication. The fundamental features of CAN bus are introduced below followed by a more detailed yet brief introduction to the topic.

Carrier Sense Multiple Access (CSMA): Each node on a bus must wait for a prescribed period of inactivity before attempting communication. Multiple nodes can attempt access to the bus concurrently. The carrier sense feature enables decentralized synchronization to the bus while multiple access simplifies arbitration.

Collision detection (CD): Collisions are detected through bit-wise logic and differential signalling. A collision is detected when a node tries to transmit a logic one and reads a logic zero. A node that detects a collision aborts communication.

Arbitration on Message Priority (AMP) Communicating nodes will output the CAN-ID first. If a collision is detected the node with the highest priority wins the arbitration and keeps transmitting.

The use of differential drivers as shown in Figure 4.2.a increases the signal to noise ratio of the access media which improves bus robustness. In the case of CAN bus, a logic zero in the transmitting side (i.e. `Tx=0`) is converted into a differential or dominant voltage in the shared data lines (i.e. `CANL=Vl`, `CANH=Vh`). A logic one in the transmitter input puts the output drivers in high impedance. When all transmitting drivers connected to the bus are in high impedance the data lines in the shared interface become a recessive level. Note that `CANL`, `CANH` take on the same voltage level driven by pull resistors (not shown here). A recessive level will be overridden by any transmitter driving a dominant bit on the bus.



a) Simplified driver

b) Arbitration example

SOF	Arbitration		Control			Data	CRC	ACK	EOF
	ID	RTR	IDE	r0	DLC				
1	11	1	1	1	4	0-8 bytes	16	2	7

c) Data frame

Figure 4.2: CAN bus fundamentals.

The differential CAN-bus transceiver in a) outputs the signal pattern as shown in b). A Standard CAN-bus frame is shown in c) where the field width is given in bits. The bubble with a '1' shows the instant where `TxB` loses arbitration.

Bus arbitration is achieved by sending the CAN arbitration field right after the start of frame (`SOF`) bit. The identifier subfield (`ID`) is transmitted first commencing from the most significant bit i.e. `ID_10`. If two nodes start transmitting concurrently as shown in Figure 4.2.b the active nodes will keep sensing the data lines to confirm that the intended output is present on the shared lines. If a node wanting to transmit a recessive bit senses a dominant bit it will stop transmission immediately due to a collision resolved in favour of a highest priority node. In this scenario the node that drives the latest dominant bit wins arbitration and continues to transmit the rest of the frame. The scheme described up to this point enforces a priority order in which `ID=0` has the highest priority

From the perspective of scheduling theory CAN bus is non preemptable because transmitting nodes cannot be interrupted. However, low priority nodes can block high priority nodes if they start transmitting before a high priority node needs access to the bus. Collisions do not affect the highest priority node among the set of nodes starting concurrent transmission. Therefore, high priority nodes will suffer less interference in the presence of spurious nodes injecting messages on the bus. As node priority decrease a high collision rate will severely impact the chances of transmission for low priority nodes. When a node normal transmission timing is affected due to blocking or high collision rates, they will need to try multiple times and that in turn will affect the inter-arrival times sequence (IATS) it generates.

4.2.2 Attack scenarios

There are two attack scenarios: (i) fuzzy injection, and (ii) impersonation or spoofing. The fuzzy injection attack, reported in the `can_fuzzy` trace, contains the longest anomalous segment and the lowest injected message count. In terms of time length and message composition, both impersonation traces (`can_gear` and `can_rpm`) are quite similar. For all three traces, the ratio of injected/normal messages is close to 0.2. Hence we assume that each experiment follows different message injection rates, which can have an impact on anomaly detection performance. More details on each attack strategy are presented below:

- Fuzzy injection: In this scenario, a fuzzy randomizing algorithm generates target `CAN-IDs`. Target CAN nodes receive unwanted random data. The attack tries to disrupt the scheduling of the bus while hindering detection. Low priority nodes should be particularly vulnerable due to CAN bus collision handling mechanism. Corrupt data might be hard to detect if it falls within the data range of the transmitting node. We did not confirm the authenticity of the target CAN nodes. The difficulty in detecting these schedule-disruptive attacks comes from the fact that each CAN node receives false messages without a discernible pattern.

- Impersonation or spoofing attacks: In this scenario, messages are injected into specific CAN nodes. Considering the attack mode tackles a single node, it could be hard to detect without knowing the ID of the target node. The nodes under attack are CAN-ID = 0x43F for the can_gear trace and CAN-ID = 0x316 for the can_rpm trace. Due to CAN priority and collision handling mechanism, this scenario should not disrupt the operation of nodes with higher priority than the node under attack. However, we found that it is disruptive to the scheduling of the bus, and its effects can be detected by analyzing high priority nodes.

Table 4.1: Overview of the HCRL-CAN dataset

Trace	Message Count $\times 10^3$				Time duration $[s]$		
	No attack	Attack		Total	No attack	Attack	Total
		Normal	Injected				
can_fuzzy	891	2456	492	3839	482	2466	2948
can_gear	891	2955	597	4443	482	1949	2431
can_rpm	891	3076	655	4622	482	1952	2434

4.2.3 Training

Each trace in the HCRL CAN injection dataset has a section of 482 s of normal data, the msg_f column differentiates between normal and injected messages. To train the model, we used window segments of 5 s taken from the normal segments of each trace. Each normal window of five seconds in duration contains around 9,250 CAN messages generated by 26 CAN IDs.

The header of the CAN trace is $H = \langle \texttt{time\_stamp}, \texttt{can\_id}, \texttt{dlc}, \texttt{data}, \texttt{msg\_f} \rangle$. Applying Definition 1 we have: $t = \texttt{time\_stamp}$, $G = \texttt{can\_id}$, and $S = \texttt{can\_id}$, i.e. the signal identifier is the generator itself. The data, dlc, and msg_f columns are excluded from the trace model.

Each normal section contains messages from 26 CAN IDs. For training, we partitioned each clean segment into windows covering five seconds of CAN data. From approximately 290 windows of normal CAN data we took a random sample of 60 windows to train the model. Then we ran MuSADET in training mode to obtain a model of the system containing all the features for each of the 26 generators. From the analysis of these features, we found that all generators are periodic, and therefore anomaly detection would be performed based on the DCTPR feature. The summary of this analysis is presented in Table 4.2.

The results recorded in Table 4.2 show that, when the system operates under normal conditions, all 26 CAN nodes send messages with a highly consistent rate. Using the median as a criterion, we concluded that all CAN nodes are likely to transmit messages at a constant rate because the DCTPR feature is above 0.95 for most window segments. The periodic behaviour is further confirmed by considering the number of traces with DCTPR above the discrimination threshold of 0.9. For example, generator `0x2C0` has the smallest minimum DCTPR with 58 windows having $D \geqslant 0.9$ an indication that `0x2C0` generates messages at a constant rate. Notice that a small number of windows scored DCTPR values below the threshold, probably due to one or two trace segments containing initialization data.

4.2.4 Anomaly detection setup

The ratio of injected messages to time duration r_a of each attack section in the HCRL CAN injection dataset is a measure of the prevalence of attacks. The variation of r_a indicates that each attack scenario follows different injection rates. For example, the `can_fuzzy` trace with $r_a = 0.199$ suggest that the fuzzy attacks are more spread than the impersonation attacks found in the `can_rpm` dataset with $r_a = 0.335$. From the dataset description we know that there are 300 localized attacks on each dataset with different injection rates. Our goal is to detect attacks as soon as possible with the best detection rate. We achieved this goal by looking into the score for each individually trace segment (window). Because the time-span of the window under analysis determines the time to detection, it is desirable to have the smallest possible window.

We tried three different window lengths: 250 ms, 500 ms, and 1 s. Two factors determined the length of the shorter window. First, the period of the fastest generators (10 ms), and second the criterion that a window is anomalous if it contains at least one injected message. For the case that an anomalous 250 ms window has one anomalous message the ratio of anomalous to normal messages is particularly small and therefore, the anomaly would be hard to detect. For example, the ratio of anomalous to normal messages for any 10 ms generator would be $^1/_{26} \approx 3.9\%$ (assuming there are precisely 25 normal messages for the generator under analysis). That is, we want to assess if MuSADET can find timing disruptions with a 3.9% minimum impact factor on any fast generator. As the window length increases the ratio of anomalous to normal messages for most anomalous windows will increase because more injected messages are likely to be present in the anomalous window.

We ran MuSADET in anomaly detection mode for the whole segment of anomalous data. We did not select a random sample of windows for anomaly detection because the anomalous segment contains enough normal data. The results of the anomaly detection experiments are presented in the next section.

Table 4.2: Training Results Summary for the HCRL-CAN dataset

CAN-ID	DCTPR			Count	Average Period [ms]			
	Min	Median	Max	D $\geqslant$0.9	normal	gear	rpm	fuzzy
0x002	0.919	0.987	0.991	60	10	12	12	18
0x0A0	0.963	0.999	0.999	60	100	121	116	183
0x0A1	0.963	0.999	0.999	60	100	122	117	188
0x130	0.891	0.986	0.992	59	10	12	12	19
0x140	0.890	0.986	0.992	58	10	12	12	19
0x153	0.924	0.995	0.997	60	10	12	12	19
0x18F	0.905	0.998	0.999	60	10	12	11	19
0x1F1	0.964	0.999	0.999	60	20	24	23	37
0x260	0.891	0.998	0.999	59	10	12	12	19
0x2A0	0.873	0.998	0.999	59	10	12	12	19
0x2C0	0.845	0.999	0.999	58	10	12	12	19
0x316	0.914	0.996	0.999	60	10	12	2	18
0x329	0.855	0.997	0.999	59	10	12	12	19
0x350	0.875	0.999	0.999	59	10	12	12	19
0x370	0.881	0.999	0.999	59	10	12	12	19
0x430	0.970	0.999	0.999	60	20	23	23	36
0x43F	0.905	0.999	0.999	60	10	3	12	19
0x440	0.896	0.999	0.999	59	10	13	12	19
0x4B1	0.974	0.999	0.999	60	20	24	23	37
0x4F0	0.926	0.999	0.999	60	20	25	24	37
0x545	0.919	0.997	0.999	60	10	13	12	20
0x5A0	0.889	0.999	1.000	58	999	1260	1215	1635
0x5A2	0.889	0.999	1.000	58	999	1281	1224	1698
0x5F0	0.956	0.999	1.000	60	50	64	61	99
0x690	0.936	0.999	0.999	60	100	124	119	189

4.2.5 Discussion

The plots in Figure 4.3 and Figure 4.4 present ROC curves for three selected nodes from the HCRL CAN dataset. Each ROC curve has three plots with data corresponding to classification scores for windows of 1 s, 500 ms, and 250 ms where each of the marked data points over the plot lines represents a classification threshold.

HCRL_GEAR_DRIVE

The ROC analysis for the spoofing scenario to the gear drive is shown in Figure 4.3. The target node for the spoofing attack is `0x43F` (centre plot), showing a positive detection rate of 100 % for windows of 1 s and 500 ms. Positive classification falls 1 % when the window length is reduced to 250 ms. In all cases, the false positive rate depends on the threshold value, where values close to $th = 0.9$ reduce false positives down to 2 %. When looking to node `0x002`, we can see that more than 75 % of anomalous windows can be detected with a low number of false positives. A similar situation, but with even better positive classification, can be observed for node `0x545` and window lengths of 1 s and 500 ms.

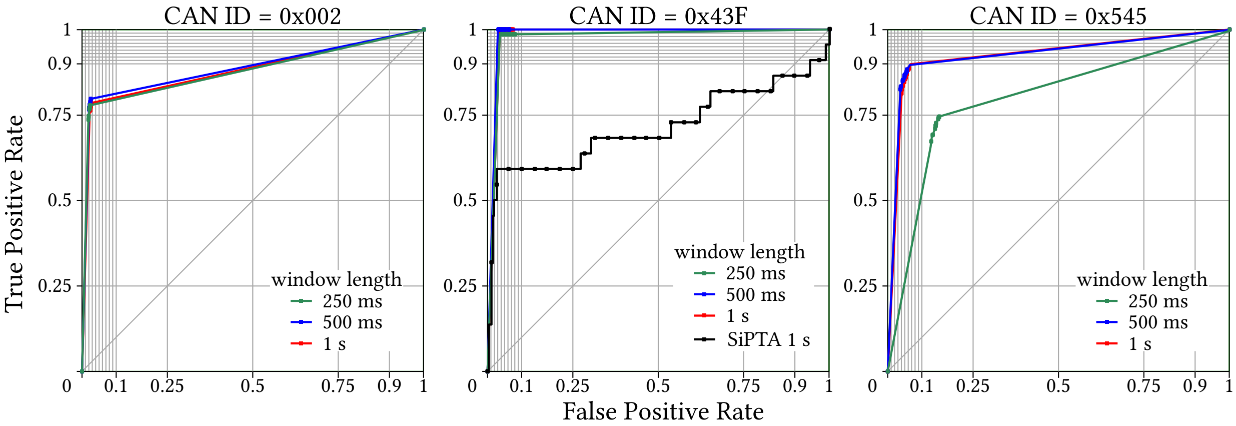


Figure 4.3: ROC curve for the `HCRL_GEAR_DRIVE` scenario.
From left to right priority on the CAN-BUS decreases. Node is `0x43F` the target of the spoofing attack.

The ROC plots in Figure 4.3 show that for this attack strategy, MuSADET can detect timing anomalies across the system with high confidence. For example, when an attack is launched, it disrupts the IATS of node `0x002` (the highest priority node), and the anomaly can be detected indirectly. The performance degradation seen for node `0x545` and window length of 250 ms is due to deep disruption of the IATS at this level of time resolution. This effect can be appreciated more clearly in the visualization plot (Figure 4.5).

Table 4.3 contains the performance analysis of selected threshold levels for MuSADET. The target node is `CAN_ID = 0x43F`; therefore, we should expect superior MuSADET's performance for this node. Table 4.3 contains three groups of classifications, one for each window length. For each window, we present performance measures for three chosen signals to show high and low priority nodes in addition to the target node. The analysis of SiPTA is not included here because from the ROC curve, and it is clear that SiPTA's performance is inadequate.

The first group of measures to analyze are precision, recall, and accuracy. For example, the precision for the 1 s window is high regardless of the node. That is an indication of a relatively low false-positive rate almost independent of node priority. As expected MuSADET's better precision is achieved for the highest priority node. This precision may seem odd, but consider that CAN bus is a prioritized and shared real-time interface. Then we can conclude that high priority nodes will generate IATS with much less variance around their expected direct current (DC) value than low priority nodes. The reduction of the IATS variance reduces the number of signals being classified as anomalous when node priority is high.

Recall and accuracy behave as we expected for injection attacks to a single node. The best results are achieved for the attacked node because it is the one that suffers the most interference. The high priority node is more immune to disruption than the low priority node. Consequently, all three measures are better for the lower priority node. As the detection window narrows, the measures do not change significantly. We attribute this behaviour to the constant ratio of normal to anomalous windows that preserve the overall distribution of Normal/Anomalous windows. A satisfying result is that MuSADET can identify all anomalous windows for the target node ($Rc = 1$), but it can also indirectly detect anomalies when analyzing low and high priority nodes.

The F1-score shows that regardless of the signal, MuSADET is capable of detecting timing anomalies with high performance, particularly for the target node. Narrowing the window does not seem to affect performance significantly. However, for the 250 ms window, the F1-score gets close to 0.9. The problem of the F1-score is that it is biased towards positive classification and does not provide a fair analysis, mainly when the two classes are not balanced. Although in our case study, both classes contain about the same amount of normal and anomalous the bias towards positive classification has some effect. We also provide the MCC score, which includes all types of predictions and therefore is more accurate. Again we can confirm that MuSADET performs well both within a particular detection window and across different windows.

Table 4.3: Summary of performance metrics for the CAN_GEAR scenario.

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.9717	0.7843	0.2157	0.8711	0.8680	0.7612
0x043F	0.9405	1.0000	0.0000	0.9658	0.9693	0.9330
0x0545	0.9439	0.8841	0.1159	0.9090	0.9130	0.8197

Window length: 1000ms Threshold=0.9

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.9668	0.7971	0.2029	0.8890	0.8738	0.7879
0x043F	0.9363	1.0000	0.0000	0.9672	0.9671	0.9365
0x0545	0.9358	0.8801	0.1199	0.9132	0.9071	0.8271

Window length: 500ms Threshold=0.9

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.9650	0.7790	0.2208	0.8869	0.8621	0.7808
0x043F	0.8603	1.0000	0.0000	0.9262	0.9249	0.8626
0x0545	0.9346	0.8726	0.1251	0.9154	0.9026	0.8298

Window length: 500ms Threshold=0.9

HCRL_FUZZY

The ROC analysis for the fuzzy injection scenario is shown in Figure 4.4. There is no target node for this scenario because messages are injected into random CAN ID's. The fuzzy injection strategy is more challenging than spoofing. From the perspective of a CAN node, these types of attacks are perceived as a moderate increase in traffic. The result can be clearly seen in the ROC curve as a decrease in positive detection when compared to target nodes in the spoofing scenarios. A characteristic feature of the fuzzy attack is a progressive disruption of the timely operation of the CAN bus as the node priority decreases.

From the ROC curve, it is evident that a reasonable amount of anomalous windows escape detection. For example, more than 25% of the anomalous windows go undetected for node `0x002`.The decrease in FNR correlates with the expectation that low priority nodes are more vulnerable to injection attacks than high priority nodes.

Since messages injected to CAN bus in the fuzzy strategy do not follow any discernible

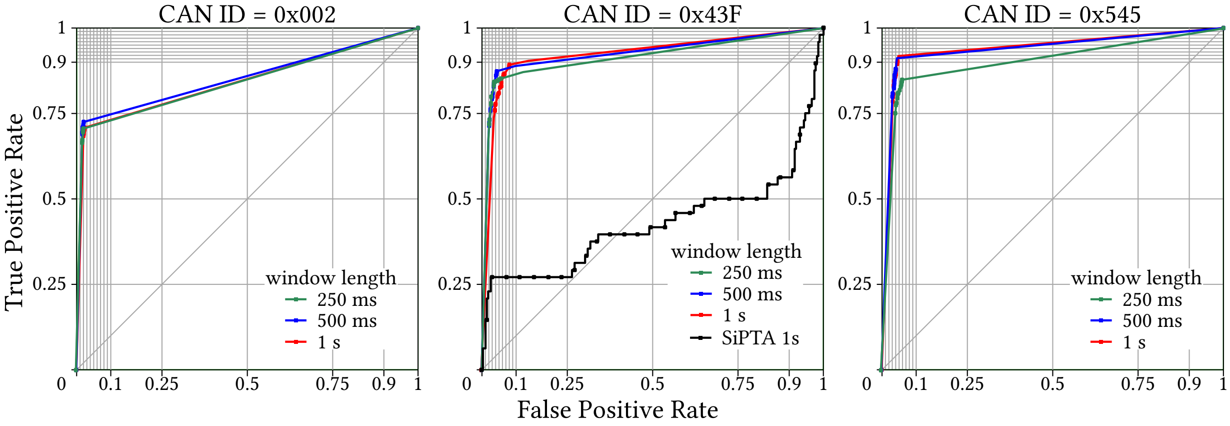


Figure 4.4: ROC curve for the HCRL_FUZZY trace.

From left to right priority on the CAN-BUS decreases. All nodes in this scenario receive injected messages.

pattern we cannot expect that for this scenario the measures will be strongly correlated to the attack mode. We would expect that the measures are somehow correlated to node priority as a result of the real-time nature of CAN bus. For example, precision follows somehow the same pattern as in the previous case, it is better for the highest priority node and then it wanders for the other two nodes. However, the mechanism by which this behaviour is repeated is due to different conditions. For the highest priority node the low variance of IATS is the key factor in avoiding misclassification of normal windows. However as node priority decreases and recall grows it becomes the dominant factor that affects the precision.

Recall and accuracy are now inversely correlated to node priority; we attribute this behaviour to an increase in CAN bus disruption as the node priority decreases. Interestingly as the detection window narrows, recall and precision increase for each node. This does not follows the pattern of injection scenarios; where these measures did not change significantly as the detection window narrowed. There is a simple explanation for this behaviour. With a wider detection window the density of anomalies within the window should be smaller. The amount of normal data in the IATS will *wash out* the anomalies making anomalous windows more likely to be classified as normal. As the detection window shortens, anomalies become much more evident and therefore detectable. In contrast, the spoofing class of attacks sends messages at a high rate to the same node and the influence per window remains unchanged as we move across the range of CAN nodes.

The F1-score, and MCC-score also correlates with node priority and detection quality is still high despite the fuzzy injection being a more challenging class of attack. Narrowing the window does not seem to affect performance between 1 s, and 500 ms with some reduction in the 250 ms. Again we can confirm that MuSADET performs well both within a particular detection window and across different windows.

Table 4.4: Summary of performance metrics for the CAN_FUZZY secenario.

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.962	0.708	0.187	0.893	0.816	0.790
0x0316	0.916	0.898	0.077	0.921	0.907	0.841
0x0545	0.951	0.919	0.064	0.944	0.934	0.889

Window length: 1000ms Threshold=0.9

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.972	0.732	0.107	0.936	0.835	0.871
0x0316	0.950	0.863	0.057	0.949	0.905	0.897
0x0545	0.955	0.910	0.041	0.958	0.930	0.916

Window length: 500ms Threshold=0.9

		Recall	Miss rate	Accuracy	Scores	
ID	Precision	(TPR)	(FNR)	(ACC)	F1	MCC
0x0002	0.960	0.709	0.105	0.935	0.816	0.864
0x0316	0.962	0.840	0.071	0.950	0.897	0.898
0x0545	0.926	0.811	0.060	0.940	0.865	0.876

Window length: 500ms Threshold=0.9

Visualizing test results

Visualization plots help identify where anomalies are located in the trace and what signals are affected by it. We present and discuss a grid plot example of classification scores for the HCRL dataset shown in Figure 4.5.

The x-axis of Figure 4.5 is the time in seconds and the y-axis CAN IDs from the `HCRL_GEAR_SPOOFING` trace. All the signals shown in Figure 4.5 are classified using the DCTPR method. Hence scores are in the interval $[0, 1]$ where values close to zero means the window under analysis is normal and close to one anomalous. We selected signals covering a wide range between one of the highest priority nodes (0x002) and the attacked node (0x43F). Small regions spanning over 100 s in the normal and anomalous portions of the trace are shown in the top plot, the bottom contains selected regions of 25 s taken from the plot above. Each cell represents the score for a window covering a fixed amount of time. Windows are 1 s long for the top plot and 250 ms for the bottom plot. Green cells show normal windows for their corresponding signal with eight levels representing the quality of the normal score. Red cells represent an anomalous score with more fine-grained levels.

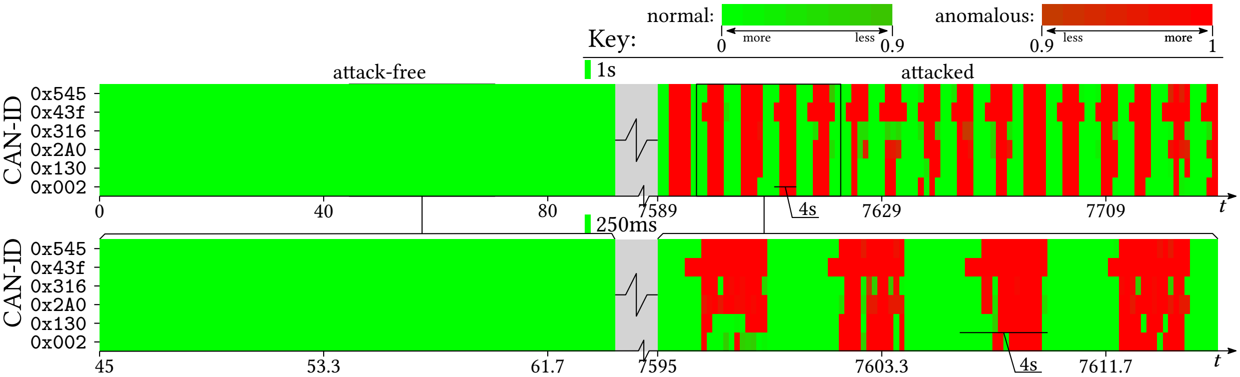


Figure 4.5: Grid of classification scores for selected CAN-ID's

Shown data is from the HCRL_GEAR_SPOOFING trace. Top-left, 100 windows of one second length in the normal segment of the trace, in the top-right, the same configuration for the anomalous region. Bottom plot is a close up of the marked regions in the plot above, each covering 25 seconds with a window length of 250 ms.

According to the description given by HCRL, each trace contains attacks lasting between 3 and 5 seconds, which can be seen in the top plot of Figure 4.5 as the vertical red areas in the anomalous segment. Also, note that anomalies are usually detected first for the attacked CAN ID (0x43F). Since the CAN specification forces nodes with lower priority to pull out of the shared bus when a collision is detected, nodes with higher priority than 0x43F should not suffer such a degree of disruption due to injected messages to node 0x43F. However, it was surprising to find that message injection could affect most signals with higher priority than 0x43F. We attribute these phenomena to the disruption of the timely operation of all other nodes in the system that produces a large number of collisions as the scheduling of the CAN bus is disturbed.

The bottom plot of Figure 4.5 shows how the detection engine performs when the window size is changed, note the factor is 1:4 when compared to the plot above. Not only detection performance remains high; the granularity also helps to pinpoint the start of the message injection process. Expected behaviour can be seen, for example; node `0x43F` *sees* the interference up to 750 ms before other nodes. For some attacks, high priority nodes (e.g., `0x002`) may suffer less interference, meanwhile as priority decreases nodes will always perceive the effects of messages injected to `0x43F`. Time to detection is also reasonable if the method is applied to online detection because attacks can be spotted and confirmed within a fraction of a second and therefore corrective measures can be taken before the attack could jeopardize the integrity of the vehicle.

MuSADET vs SiPTA

The centre plots in the ROC curves (Figure 4.3 and Figure 4.4) shows the performance of SiPTA for a window length of one second. The positive detection rate of SiPTA is close to 60% in the best case. There are several reasons why SiPTA performs so poorly compared to MuSADET. First SiPTA aggregates scores for all its channels into an overall score for the whole window and therefore we cannot provide per-signal analysis. For the `can_gear` scenario where the attack is launched to a single node the effect of aggregating signal scores cannot pinpoint that node `0x43F` has been attacked. Another reasons come from assumptions made by SiPTA and the choice of classification feature. In SiPTA scores are based on the maximum non-DC amplitude peak of the Fourier Transform. Since the IATS for the HCRL CAN dataset are all DC sequences SiPTA's assumption does not hold and therefore SiPTA is unsuitable for this type of scenario. Finally we could not run SiPTA for window lengths smaller than one second because the IATS are too short and they would not provide enough information for accurate classification. Due to these factors, it is clear why SiPTA is unsuitable for anomaly detection when IATS are DC.

4.3 QNX HEXACOPTER

Traces in this dataset where collected by the Embedded Systems Group at the University of Waterloo, Canada. The QNX operating system (QNX) Flight Control dataset [66] contains recorded events for a gyro-stabilized Mikrokopter hexacopter. There are nine normal traces in the dataset and a set of traces where different interfering strategies were tested trying to produce faults. In the interfering group five traces are known to have produced timing anomalies into the hexacopter. All traces cover around 16 seconds of operation. Interference processes were launched as bash scripts by a user with administrative privileges.

Table 4.5: Overview of the QNX HEXACOPTER dataset

Trace name	Anomaly class	N/A Windows
clean_02 ··· clean_10	`normal`	all/0
fifo_ls_sporadic	`fifo`	6/7
fifo_ls_01	`fifo`	6/7
fifo_ls_02	`fifo`	5/8
hil_full_while	`while_loop`	—/all
hil_half_while	`while_loop`	—/all

There are two types of anomalies in this dataset. The `fifo` group tries to emulate a livelock scenario as the run script launches two processes that compete with other processes at the same priority level and slow overall system's progression over time. The `while_loop` scenario emulates process starvation by launching a while loop that consumes as much computing power as possible.

The summary of the HEXACOPTER dataset is shown in Table 4.5. Two types of interference are introduced into the HEXACOPTER during operation. In the `fifo` group, the anomaly consists of executing a script where an interfering process, launched sporadically, runs the `ls` command. In `while` group, the interfering process launches a script that runs a while loop continuously. No labels are provided in the original dataset. Therefore it would be impossible to run ROC analysis for these traces unless assuming that the whole trace is anomalous. We overcome this issue by identifying when the interfering process was launched and annotating the windows accordingly.

4.3.1 Training

Each trace in the HEXACOPTER dataset contains around 16 seconds of data. In some traces initialization data is present for up to three seconds and up to 13 seconds of useful data for training and anomaly detection. Traces in this dataset are fairly shorter compared to the HCRL CAN dataset. We opted for 2 s training windows, and were able to divide the training set into 67 windows of which 33 were randomly selected for the model.

The header of the HEXACOPTER traces contains a large number of columns but only a few can be used for generators and signals. After applying Deffinition 1 we have: $t = \texttt{time\_stamp}$, $G = \texttt{PID}, \texttt{TID}$, and $S = \texttt{CLASS}, \texttt{EVENT}$, where:

`PID`: Process identifier.

TID: Thread identifier.

CLASS: Where the event originates, for example THREAD is a message originated in the thread.

EVENT: Type of event, for example THREPLY is a reply sent by the process in response to some received message.

After running MuSADET in training mode we identified around 140 signals in this dataset. In some cases there were more depending on whether the system produced sporadic events or some initialization data was still present in some training windows. Of the total of signals found many did not meet the requirements of MuSADET, for example, the IATS of sporadic events do not have samples for analysis. Some other events are still recurrent but of too low frequency to be useful and while some signals where identified as DC we already covered that especial case in the previous section. We finally chose a subset of suitable signals to discuss anomaly detection results. In particular we used signals that become too short when the detection window is reduced, signals that do not indicate the presence of anomalous behaviour and signals that can be used to detect anomalies for the scenarios available in the dataset.

4.3.2 Anomaly detection setup

From the model we selected a subset of signals all non classifiable by DCTPR. We processed all test traces containing anomalies with MuSADET and SiPTA but since the number of normal windows present in the anomalous traces is small we also took a random sample of normal windows from the normal set. By incorporating more normal windows from the normal set we were able to balance the two classes.

We ran MuSADET and classified the set of test windows by using the power spectral density (PSD) and the binary power spectral sequence (BPSS) features along with their corresponding distances. For SiPTA we used the same subset of signals as channels and ran the classifier. The results of this process is discussed bellow.

4.3.3 Discussion

The ROC curves for the HEXACOPTER dataset are shown in Figure 4.6, Figure 4.7, and Figure 4.8. Each figure contains plots for the same signals but different groups of traces. For example, Figure 4.6 covers anomaly detection by applying the χ^2 symmetric distance to PSD features, while Figure 4.7 applies the Jaccard distance to BPSS. The left plot contain ROC curves for the combination `fifo and while loop` and the other two plots analyze each type of interference independently. Since MuSADET does not have an algorithm to tune the detection threshold combining both types of interference provides insight on how MuSADET performs in a more complex environment.

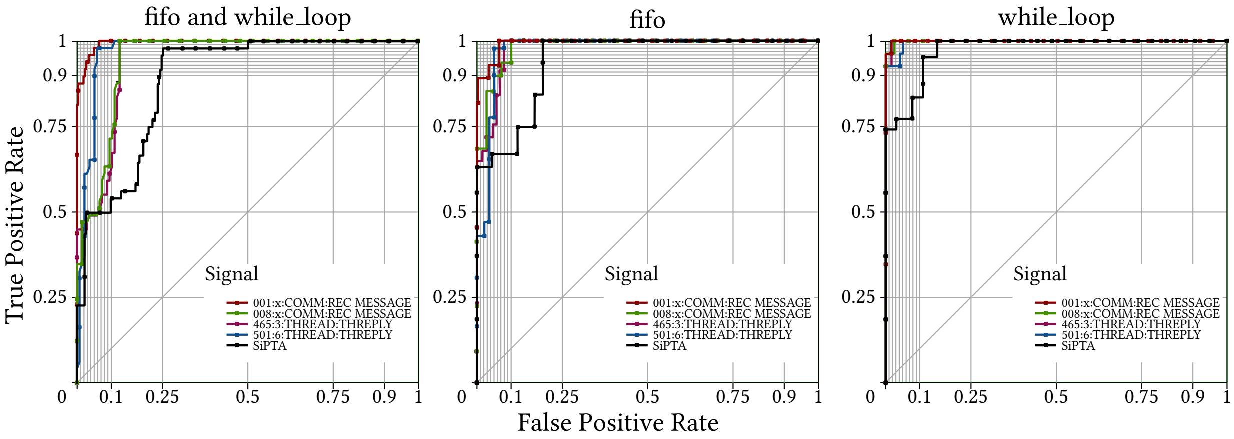


Figure 4.6: ROC curves for the QNX HEXACOPTER dataset.
Classification scores of MuSADET are based on the PSD and the χ^2-symmetric distance.

The left plot of Figure 4.6 shows that for two of the selected signals positive detection rate is above 98 % with a false positive rate below 7 %, all other signals, excluding SiPTA also show good classification accuracy. The centre and right plots present the same analysis for each scenario. Note that MuSADET can detect both types of anomalous behaviour with almost exact classification in some cases. The split ROC analysis demonstrates that MuSADET is robust when facing different anomalous modes but also suggests that classification thresholds are anomaly-dependent. For example, starvation or greedy consumption (e.g., `while_loop` scenario) of resources seems easier to detect than computation demand that slows down overall temporal progression (e.g., `fifo` scenario).

The ROC curves of Figure 4.7 show a striking resemblance to Figure 4.6. This plots correspond to classification by Jaccard distance applied to BPSS. The main goal when using the BPSS is to reduce computational load and reduce the effect of non dominant peaks when classifying the signals. The intuition behind BPSS being that the frequency domain (FD) of IATS generated by recurrent generators will have dominant peaks that can be used as the defining feature for anomaly detection. That rationale is behind SiPTA and proved to be correct even when it makes assumptions that do not hold for all cases.

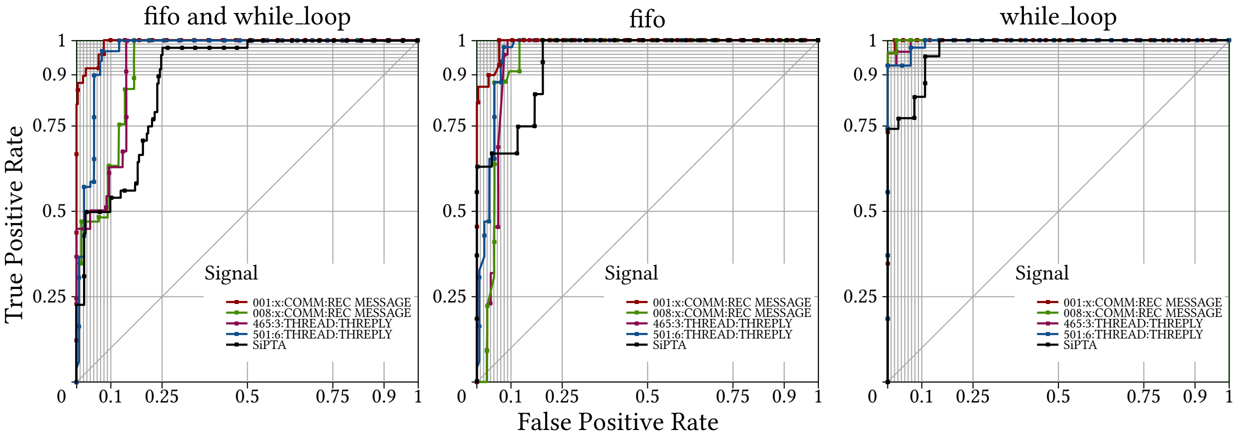


Figure 4.7: ROC curves for the QNX HEXACOPTER dataset.
Classification scores of MuSADET are based on Binary Power Spectral Sequence feature and Jaccard distance.

As we expected the use of BPSS would produce better results than SiPTA because we are using more FD information to compute the scores. Another advantage of MuSADET is classification at signal level without further aggregations that can let anomalies undetected.

A better comparison between classification using PSD and BPSS is presented in Figure 4.8. As expected the PSD feature outperforms BPSS for the same reason stated above. The PSD feature contains more information from the frequency spectrum than BPSS and therefore predictions based on the former should be more accurate. Despite this difference, we conclude that BPSS is a feature that can be effectively used for anomaly detection in the context studied in this thesis.

The measures for PSD and BPSS are reported in Table 4.6 and Table 4.7 respectively. The results demonstrate that anomaly detection by applying FD features is feasible. Note that in all cases recall is high, hence all methods are capable of retrieving anomalous windows. However it also comes with drawbacks, the first one being relative lack of precision, an indication that the false positive rate is relatively high. Overall performance in terms of correct detection is moderate with an accuracy close to 0.9 for all reported signals, except for SiPTA. That is MuSADET correctly classifies signals with a 90% hit rate.

A better measure to capture the relation between precision and recall is the F1-score which becomes more relevant for the HEXACOPTER dataset because the classes are not balanced like in HCRL. There are more normal than anomalous windows in the test set. Moreover the F1-score gives is based on TP, FP, and P hence it focuses on the quality of positive predictions by taking into account the wrong positive predictions. For example,

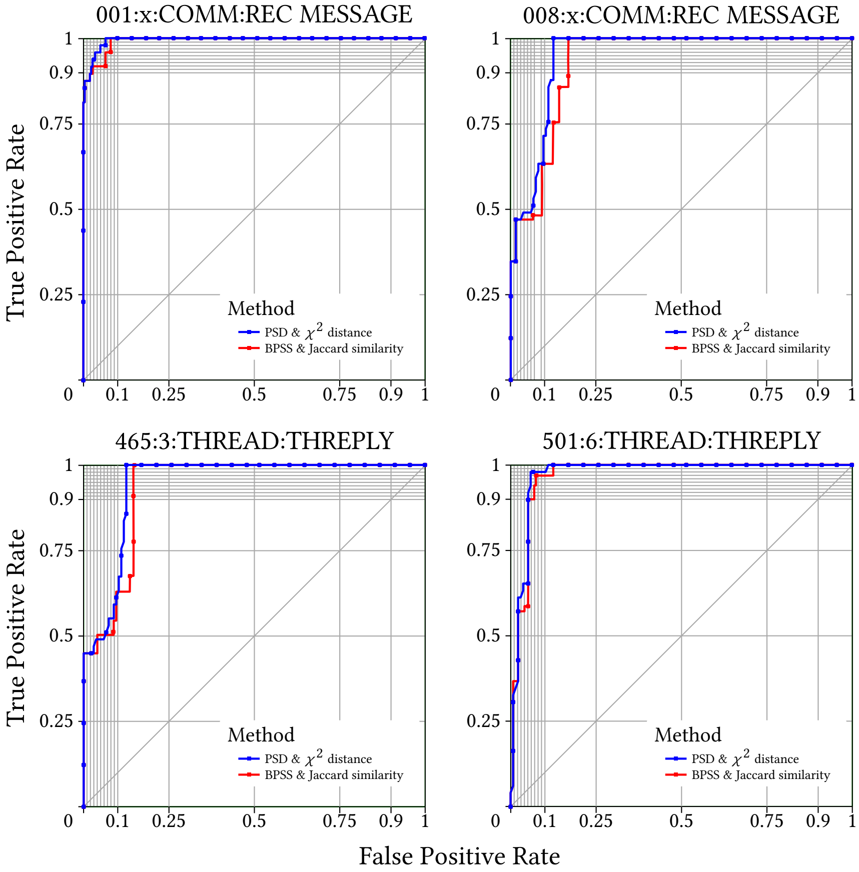


Figure 4.8: ROC curves for the QNX HEXACOPTER dataset.
Each plot contains the ROC curve based on classification from PSD and BPSS features.

for most signals reported, MuSADET can recall all anomalous windows, but it does so by also rising false alarms. For most cases MuSADET's F1-scores are close to its accuracy which in this case is an indicator of good performance. After incorporating the MCC measure we can conclude that there is also a good correlation between the true values and the classification for both the anomalous and normal classes. Hence, despite differences in class sizes we do have a classifier that is accurate and unbiased.

Table 4.6: Performance metrics for the HEXACOPTER. Classification by PSD.

	Precision	Recall	Miss rate	Acc	Scores	
Generator:Signal	(PPV)	(TPR)	(FNR)	(ACC)	F1	MCC
001:COMM_REC_MESSAGE	0.742	1.000	0.041	0.939	0.852	0.811
008:COMM_REC_MESSAGE	0.590	1.000	0.061	0.881	0.742	0.684
465_3:THREAD_THREPLY	0.590	1.000	0.041	0.884	0.742	0.695
501_3:THREAD_THREPLY	0.762	0.980	0.102	0.935	0.857	0.793
SiPTA	0.420	0.959	0.184	0.761	0.584	0.467

Window length: 1000ms Metric: Simmetricχ^2

Thresholds: MuSaDET = 8.5, SiPTA = 0.002

Table 4.7: Performance metrics for the HEXACOPTER. Classification by BPSS.

	Precision	Recall	Miss rate	Acc	Scores	
Generator:Signal	(PPV)	(TPR)	(FNR)	(ACC)	F1	MCC
001:COMM_REC_MESSAGE	0.700	1.000	0.041	0.926	0.824	0.780
008:COMM_REC_MESSAGE	0.538	1.000	0.102	0.848	0.700	0.619
465_3:THREAD_THREPLY	0.563	1.000	0.061	0.868	0.721	0.661
501_3:THREAD_THREPLY	0.716	0.980	0.143	0.916	0.828	0.741
SiPTA	0.420	0.959	0.184	0.761	0.584	0.467

Window length: 1000ms Metric: Simmetric BPSS

Thresholds: MuSaDET = 8.5, SiPTA = 0.002

Visualizing classification scores

The plot in Figure 4.9 is similar to the previously discussed for the HCRL dataset. Figure 4.9 shows classification scores for the normal trace `hil_clean_04` while the top-right for trace `hil_fifo_ls_02` containing timing anomalies. All scores shown are for classification by χ^2 distances with normal scores in the interval $[0, 24.996)$, and anomalous in [24.996,38]. When $V(a,i) \geqslant 38$ it is set to 38 because for such a χ^2 value the corresponding p-value falls in the interval $p = (0, 0.001]$. The threshold is set to $V(a,i) = 24.996$ with $p = 0.05$ for a right-tail χ^2 distribution with 15 degrees of freedom. The bottom plots of Figure 4.9 show the same trace above processed with a window length of 500 ms.

There are six known anomalies in trace `hil_fifo_ls_02` consisting on running a bash script that executes a loop that sleeps for 1 s and then runs the command `ls`. There

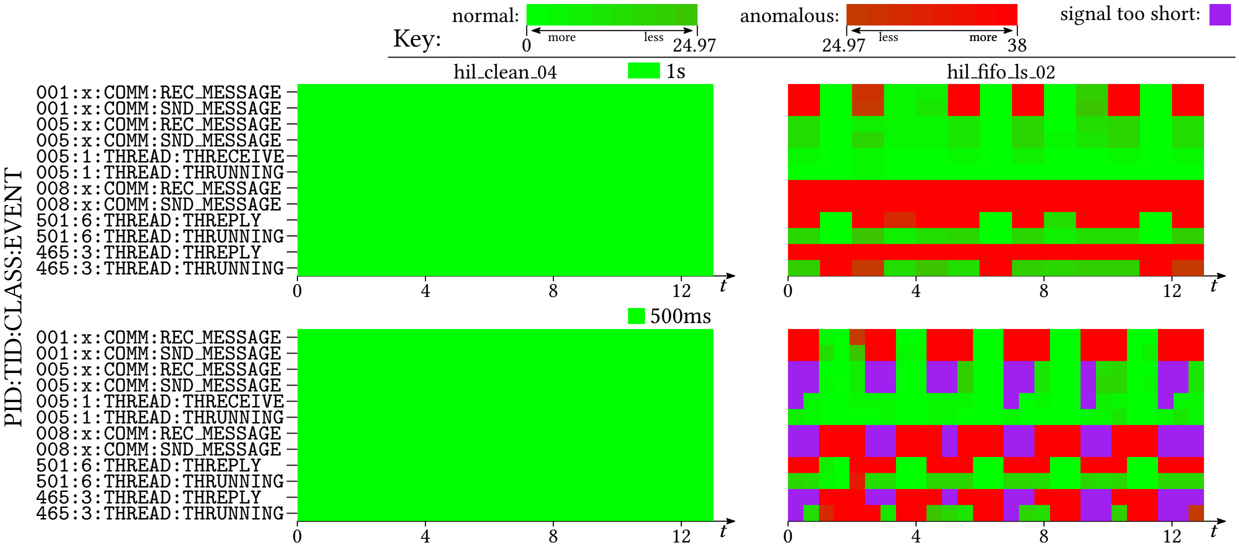


Figure 4.9: Grid of classification scores for selected signals from the HEXACOPTER dataset.

Top-left, scores for the `hil_clean_04` trace showing a normal case. Top-right, scores for the `hil_fifo_ls_02` trace showing various anomalies. Bottom plots are close ups of the corresponding plot above with a window length of 500 ms. Notice in the bottom-right plot that many windows do not contain enough inter-arrival times and hence are classified as anomalous by the `SIGNAL TO SHORT` criterion.

is no trace annotation for anomalies in the HEXACOPTER dataset, hence information about the anomalies was found by careful inspection of process activity. Each time the `ls` command is run, there is timing disruption across the system, and the six times the command ran can be seen affecting various signals. Figure 4.9 also shows that not all signals are affected by this type of anomaly. For example, `005:1:THREAD:THRUNNING` is immune to the execution of the `ls` command. Note that most affected signals are in the `COMM` class which is related to kernel-IO activity, and this supports our findings because `ls` is also IO heavily oriented. When the window resolution is increased from 1 s to 250 ms some signals that looked normal become too short to compute their PSD and therefore are classified as anomalous by the `SIGNAL TO SHORT` criterion. For example, `005:x:COMM:SND_MESSAGE` was considered normal in the 1 s analysis, but it was also close to the normal limit, then it became too short for PSD computation for a 500 ms window.

QNX HEXACOPTER MuSADET vs. SiPTA

By looking into the ROC curves it is clear that MuSADET outperforms SiPTA as a general anomaly detector (e.g., 99 % TPR and 5 % FPR MuSADET vs. 98 % TPR and 25 % FPR SiPTA). Note that SiPTA is equally effective to MuSADET for some types of anomalies. A key advantage of MuSADET is its ability to tell what signal is being affected by the anomaly. SiPTA classifies a window as a whole, and no further insight is provided about what generators and event classes are affected by the anomaly.

Chapter 5

Conclusions

Identifying unexpected behavior is crucial for safety-critical embedded systems. In this work, we demonstrated that event traces are a suitable source of information to detect anomalies such as attacks on controller area network (CAN) or unwanted computation demand. The feasibility of our approach relies on the classification of frequency domain (FD) features computed from inter-arrival times sequence (IATS), namely DC to total power ratio (DCTPR) or power spectral density (PSD) estimation, and the choice of an appropriate similarity measure to compare analysis features to model features.

The classification of direct current (DC) IATS by DCTPR yields excellent classification. When multi-signal anomaly detection for real-time traces (MuSADET) is compared to Signal Processing for Trace Analysis (SiPTA), it is clear that SiPTA relies on assumptions that do not hold when the system is mainly composed of periodic generators. Therefore DC signals should not be ignored when detecting timing anomalies nor they should be classified by PSD estimation or any aggregating method like SiPTA.

In the case of PSD estimation combined with similarity measures to compare the power spectra our anomaly detection (AD) engine detected timing anomalies that propagate throughout the system. MuSADET generally outperforms SiPTA and provides further information about the anomaly. For example, grid plots can be used to establish relationships between the anomaly and affected generators.

References

[1] M Aiello, E Cambiaso, M Mongelli, and G Papaleo. An on-line intrusion detection approach to identify low-rate dos attacks. In *Security Technology (ICCST), 2014 International Carnahan Conference on*, pages 1–6. IEEE, 2014.

[2] Neil C. Audsley, Alan Burns, M. F. Richardson, Ken W. Tindell, and Andy J Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[3] M. Bartlett. The statistical approach to the analysis of time-series. *Transactions of the IRE Professional Group on Information Theory*, 1(1):81–101, Feb 1953.

[4] M. S. Bartlett. Periodogram analysis and continuous spectra. *Biometrika*, 37(1/2):1–16, 1950.

[5] B Berckmoes, R Lowen, and J Van Casteren. Distances on probability measures and random variables. *Journal of Mathematical Analysis and Applications*, 374(2):412–428, 2011.

[6] E. Bini, T. Huyen Chau Nguyen, P. Richard, and S. K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computers*, 58(2):279–286, Feb 2009.

[7] R. B. Blackman and J. W. Tukey. *The measurement of power spectra from the point of view of communications engineering.* New York : Dover Publications, 1959.

[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[9] Christopher D. Brown and Herbert T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24 – 38, 2006.

[10] Suratna Budalakoti, Ashok N Srivastava, Ram Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. 2006.

[11] Alan Burns, Ken W. Tindell, and Andy J Wellings. Effective Analysis for Engineering Real-Time Fixed Priority Schedulers. *IEEE Transactions on Software Engineering*, 21(5):475–480, May 1995.

[12] Alan Burns and Andy J Wellings. Engineering a Hard Real-Time System: From Theory to Practice. *Software Practice and Experience*, 25(7):705–726, July 1995.

[13] Jorge Caiado, Nuno Crato, and Daniel Peña. A periodogram-based metric for time series classification. *Computational Statistics & Data Analysis*, 50(10):2668–2684, 2006.

[14] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[15] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[16] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[17] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

[18] V. Chandola, V. Mithal, and V. Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 743–748, Dec 2008.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

[21] Yu Chen and Kai Hwang. Collaborative detection and filtering of shrew ddos attacks using spectral analysis. *Journal of Parallel and Distributed Computing*, 66(9):1137–1151, 2006.

[22] Albert M.K. Cheng. *Real-Time Systems. Scheduling Analysis and Verification.* Number ISBN 0-471-18406-3. Wiley Insterscience, 2002.

[23] X. Cheng, K. Xie, and D. Wang. Network traffic anomaly detection based on self-similarity using hht and wavelet transform. In *Fifth International Conference on Information Assurance and Security, IAS*, volume 1, pages 710–713. IEEE, 2009.

[24] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[25] David Roxbee Cox, David Roxbee Cox, David Roxbee Cox, and David Roxbee Cox. *Renewal theory*, volume 1. Methuen London, 1967.

[26] Michel Marie Deza and Elena Deza. Encyclopedia of distances. In *Encyclopedia of distances*, pages 1–583. Springer, 2009.

[27] J.L. Diaz, Kanghee Kim, José María López, and L. Lo Bello. Stochastic analysis of priority-driven periodic real-time systems. In *Handbook of Real-Time and Embedded Systems*, 2007.

[28] José Luis Díaz, Daniel F García, Kanghee Kim, Chang-Gun Lee, L Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. Stochastic analysis of periodic real-time systems. In *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, pages 289–300. IEEE, 2002.

[29] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification.* Wiley, New York, 1973.

[30] Graham Dunn and Brian S Everitt. *An introduction to mathematical taxonomy.* Courier Corporation, 2004.

[31] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[32] C J Fidge. Real-Time Schedulability Tests for Preemptive Multitasking. *Real-Time Systems*, 14(1):61–93, 1998.

[33] Robert G Gallager. *Discrete stochastic processes*, volume 321. Springer Science & Business Media, 2012.

[34] Bo Gao, Hui-Ye Ma, and Yu-Hang Yang. Hmms (hidden markov models) based on anomaly intrusion detection method. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 1, pages 381–385. IEEE, 2002.

[35] J. Gao, G. Hu, X. Yao, and R. K. C. Chang. Anomaly detection of network traffic based on wavelet packet. In *Asia-Pacific Conference on Communications. APCC*, pages 1–5. IEEE, 2006.

[36] Gabriella Gigante and Domenico Pascarella. *Formal Methods in Avionic Software Certification: The DO-178C Perspective*, pages 205–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[37] Gerald Goertzel. An algorithm for the evaluation of finite trigonometric series. *The American Mathematical Monthly*, 65(1):34–35, 1958.

[38] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[39] Oleg Iegorov, Vincent Leroy, Alexandre Termier, Jean-François Méhaut, and Miguel Santana. Data mining approach to temporal debugging of embedded streaming applications. In *Proceedings of the 12th International Conference on Embedded Software*, pages 167–176. IEEE Press, 2015.

[40] Oleg Iegorov, Reinier Torres, and Sebastian Fischmeister. Periodic task mining in embedded system traces. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 331–340, Porto, Portugal, 2017.

[41] Taylor T. Johnson, Raghunath Gannamaraju, and Sebastian Fischmeister. A survey of electrical and electronic (e/e) notifications for motor vehicles. In *24th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, pages 1–15, Gothenburg, Sweden, 2015.

[42] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[43] Peter Kafka. The automotive standard iso 26262, the innovative driver for enhanced safety assessment & technology for motor cars. *Procedia Engineering*, 45:2 – 10, 2012. 2012 International Symposium on Safety Science and Technology.

[44] D I Katcher, H Arakawa, and J K Strosnider. Engineering and Analysis of Fixed Priority Schedulers. *IEEE Trans. on Software Engineering*, 19(9):6, 1993.

[45] S. Kay. *Intuitive Probability and Random Processes using MATLAB®*. Intuitive Probability and Random Processes Using MATLAB. Springer US, 2006.

[46] John Frank Charles Kingman. *Poisson processes*, volume 3. Clarendon Press, 1992.

[47] T. Kohda and Y. Takagi. Accident cause analysis of complex systems based on safety control functions. In *RAMS '06. Annual Reliability and Maintainability Symposium, 2006.*, pages 570–576, Jan 2006.

[48] Eugene F Krause. *Taxicab geometry: An adventure in non-Euclidean geometry.* Courier Corporation, 2012.

[49] Krishan Kumar, RC Joshi, and Kuldip Singh. A distributed approach using entropy to detect ddos attacks in isp domain. In *2007 International Conference on Signal Processing, Communications and Networking*, pages 331–337. IEEE, 2007.

[50] Terran Lane, Carla E Brodley, et al. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49, 1997.

[51] Philip A Laplante. *Real-Time Systems Design and Analysis. Third Edition.* IEEE Press and Wiley-Interscience, 2004.

[52] Pierre Legendre and Marti J Anderson. Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. *Ecological monographs*, 69(1):1–24, 1999.

[53] John P. Lehoczky, Lui Sha, and Ye Ding. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. In *Proceedings of the 10th IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica, CA , USA, 1989.

[54] N. G. Leveson and C. S. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, July 1993.

[55] C L Liu and J W Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[56] Haiqin Liu and Min Sik Kim. Real-time detection of stealthy ddos attacks using time-series decomposition. In *2010 IEEE international conference on communications*, pages 1–6. IEEE, 2010.

[57] José María López, José Luis Díaz, Joaquín Entrialgo, and Daniel García. Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-Time Systems*, 40(2):180, 2008.

[58] W. Lu and A. A. Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, 2009, 2009.

[59] D.G. Manolakis and V.K. Ingle. *Applied Digital Signal Processing: Theory and Practice*. Cambridge University Press, 2011.

[60] Christoph C Michael and Anup Ghosh. Two state-based approaches to program-based anomaly detection. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference*, pages 21–30. IEEE, 2000.

[61] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Pearson Education, 2011.

[62] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. Change-point cloud ddos detection using packet inter-arrival time. In *Computer Science and Electronic Engineering (CEEC), 2016 8th*, pages 204–209. IEEE, 2016.

[63] Keith Pazul. Controller area network (can) basics [an713]. Technical report, Microchip Technology Inc, 1999.

[64] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. And Intelligent Manufacturing Systems. Prentice Hall, 1996.

[65] S. Rawat and C. S. Sastry. Network intrusion detection using wavelet analysis. In *Intelligent Information Technology*, pages 224–232. Springer, 2005.

[66] Yassir Rizwan. Towards high speed aerial tracking of agile targets. Master's thesis, University of Waterloo, 2012.

[67] M. Salagean and I. Firoiu. Anomaly detection of network traffic based on analytical discrete wavelet transform. In *8th International Conference on Communications (COMM)*, pages 49–52. IEEE, 2010.

[68] Neil J Salkind. *Encyclopedia of measurement and statistics*, volume 1. Sage, 2007.

[69] Arthur Schuster. On the investigation of hidden periodicities with application to a supposed 26 day period of meteorological phenomena. *Terrestrial Magnetism*, 3(1):13–41, 1898.

[70] Richard Serfozo. Renewal and regenerative processes. In *Basics of Applied Stochastic Processes*, pages 99–167. Springer, 2009.

[71] Lui Sha, Tarek F. Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore P. Baker, Alan Burns, Giorgio C. Butazzo, Marco Caccamo, John P. Lehoczky, and Aloysius K Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28(2):101–155, 2004.

[72] Frank Singhoff, Alain Plantec, Pierre Dissaux, and Jérôme Legrand. Investigating the usability of real-time scheduling theory with the cheddar project. *Real-Time Systems*, 43(3):259–295, Nov 2009.

[73] Mikael Sjodin and Hans Hansson. Improved response-time analysis calculations. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*, pages 399–408. IEEE, 1998.

[74] Robert R. Sokal. Distance as a measure of taxonomic similarity. *Systematic Zoology*, 10(2):70–79, 1961.

[75] SteveCorrigan. Introductionto the controllerareanetwork(can). Technical report, Texas Instruments, 2016.

[76] Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.

[77] Lothar Thiele and Pratyush Kumar. Can real-time systems be chaotic? In *Embedded Software (EMSOFT), 2015 International Conference on*, pages 21–30. IEEE, 2015.

[78] Annie H Toderici and Mark Stamp. Chi-squared distance and metamorphic virus detection. *Journal of Computer Virology and Hacking Techniques*, 9(1):1–14, 2013.

[79] Charles F Van Loan. Introduction to scientific computing. the matlab curriculum series, 2000.

[80] Michail Vlachos, Philip Yu, and Vittorio Castelli. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 449–460. SIAM, 2005.

[81] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, Jun 1967.

[82] P. Welch. A fixed-point fast fourier transform error analysis. *IEEE Transactions on Audio and Electroacoustics*, 17(2):151–157, Jun 1969.

[83] N. Ye and X. Li. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, pages 171–174. Oakland: IEEE, 2000.

[84] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112, 2001.

[85] Mohammad Mehdi Zeinali Zadeh, Mahmoud Salem, Neeraj Kumar, Greta Cutulenco, and Sebastian Fischmeister. Sipta: Signal processing for trace-based anomaly detection. In *Proceedings of the 14th International Conference on Embedded Software*, page 6. ACM, 2014.

[86] M. Zhou and S. D. Lang. Mining frequency content of network traffic for intrusion detection. In *Proceedings of the IASTED International Conference on Communication, Network, and Information Security*, 2003.

APPENDICES

Appendix A

Comparison to Alternative Methods

Explanatory note

Multi-signal anomaly detection for real-time traces (MuSADET) builds on Signal Processing for Trace Analysis (SiPTA) a work that was originally published by Mehdi et al. [85]. The original paper compares the signal processing approach to alternative methods such as Markov chains and Neural networks. Despite SiPTA's having better performance than the other methods it had flaws that where addressed in an extended paper of which I was the main author. The paper was submitted to ACM Transactions on Embedded Computing Systems. The draft was rejected with comments on its fundamental flaws, particularly some related to the features used for anomaly detection and the classification metric. After thorough analysis I arrived to the conclusion that there where fundamental flaws in the trace to signal model. Also, the critics pointed to fundamental problems that would potentially render SiPTA useless if assumptions failed. Nevertheless, SiPTA still outperforms the method to which was compared and therefore it showed potential for improvement.

The comparison between SiPTA and other methods was carried out by some of the authors in the draft submitted to ACM Transactions and therefore they are not my original work. I have included below the paper submitted to ACM Transactions of Embedded Computing as an indirect comparison between SiPTA's later improvement to other methods used for anomaly detection. I was not able to conduct the experiments of MuSADET and all contributors to the original work had already graduated by the time MuSADET was complete. Since I do provide a comparative analysis between SiPTA and MuSADET the following text should serve as reference for the improvement of MuSADET over other methods.

SiPTA: Anomaly Detection in Embedded Systems Through Signal Processing of Event-Based Traces.

REINIER TORRES, MOHAMMAD MEHDI ZEINALI ZADEH, CARLOS MORENO, MAHMOUD SALEM, NEERAJ KUMAR, GRETA CUTULENCO, and SEBASTIAN FISCHMEISTER, University of Waterloo, Canada

Given a set of traces, trace-based anomaly detection tries to find anomalous behaviour. Most current approaches focus on application outside of the embedded systems domain leaving unexploited some of its properties.

This work introduces Signal Processing for Trace-based Anomaly Detection (SiPTA) a novel technique for offline trace-based anomaly detection exploiting the intrinsic periodicity of embedded systems. The method is particularly useful for embedded systems, and the paper demonstrates this by comparing SiPTA to Markov Model, and Neural Networks. This article shows the feasibility of SiPTA through case studies involving traces from a variety of embedded systems. In the experiments, our approach outperformed every other tested method.

Additional Key Words and Phrases: Anomaly detection, Embedded software monitoring, signal processing, signals and systems analysis.

ACM Reference Format:
Reinier Torres, Mohammad Mehdi Zeinali Zadeh, Carlos Moreno, Mahmoud Salem, Neeraj Kumar, Greta Cutulenco, and Sebastian Fischmeister. 00. SiPTA: Anomaly Detection in Embedded Systems Through Signal Processing of Event-Based Traces. . *ACM Trans. Embedd. Comput. Syst.* 00, 00, Article 00 (00), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Trace-based anomaly detection can be used as a monitoring and fault recovery tool for safety-critical embedded systems. Indeed, monitoring is a requirement in safety standards like ISO-26262 [19] for functional automotive, and DO-178C [16] for airborne systems. Trace-based anomaly detection (TAD) aims at detecting patterns that do not belong to the known functioning of the system. It uses time-stamped event traces as information source from which anomalous behaviour can be identified. The method proposed in this paper leverages the availability of tracing tools, signal processing, a distance metric, and a threshold based classifier to detect timing anomalies in embedded systems.

Event tracers, used mostly as logging and debugging tools, are available as an internal module in many real-time operating systems [34]. External tracers such as network loggers are also accessible, enabling the analysis of network protocols like controller area network (CAN) bus [17]. There are, at least, two types of anomalies that TAD for embedded systems can target: (i) anomalous execution sequences, and (ii) timing anomalies. We only focus on timing anomalies, those that arise when the

Authors' address: Reinier Torres, rtorresl@uwaterloo.ca; Mohammad Mehdi Zeinali Zadeh, mmzeinal@uwaterloo.ca; Carlos Moreno, cmoreno@uwaterloo.ca; Mahmoud Salem, m4salem@uwaterloo.ca; Neeraj Kumar, n26kumar@uwaterloo.ca; Greta Cutulenco, gcutulen@uwaterloo.ca; Sebastian Fischmeister, sfischme@uwaterloo.ca, University of Waterloo, ON, Canada.

1539-9087/00/00-ART00 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

execution of programs within the system follows unexpected paths that change its responsiveness. A fundamental advantage of TAD is its capacity to treat the system under scrutiny as a black box and still be able to identify anomalous behaviour. The black box approach to system's behaviour comes at the expense of classification errors and the need to train the classifier.

The fundamental challenge of trace-based anomaly detection is how to identify an incorrect behavior without raising too many false alarms. Related work in a comprehensive survey [9] remarks the importance of checking the effectiveness of detection mechanisms through *false positives*, and *false negatives*. High false alarm rates diminish the value of the tool because the users stop trusting it. False positives are even more detrimental to the effectiveness of the detector due to the overlooking anomalous behaviour. A well-known method to evaluate the quality of a detector is through receiver operational characteristic (ROC) analysis.

Detection of anomalies can be done, online (during runtime) or offline (after the program stops). Offline anomaly detection considers an entire trace of a system execution scenario for analysis, while the online approach works on streams of execution events to detect anomalies on-the-fly. Anomaly detection can also be done using supervised or unsupervised methods. Supervised methods require the training of a classification engine, and therefore a set of labelled traces is needed during training.

This paper presents SiPTA, which realizes a novel technique for offline trace-based anomaly detection that exploits the intrinsic feature of periodic processes found in embedded systems. This article is an extension of our previous work that appeared in EMSOFT [41]. As we explain later, SiPTA uses signal processing algorithms to identify periodic features in times series extracted from event traces recorded during embedded systems operation. The method presented in this extended version of SiPTA improves our previous work presented in [41], the contributions of this paper are outlined as follows:

Exploits the intrinsic periodicity of embedded systems task sets. Embedded systems are usually implemented as a set of periodic tasks (programs or processes), and we take advantage of this feature to extract inter-arrival time series (IATS) from event traces. Instead of analyzing IATS applying approaches found in most anomaly detection methods [10] we compute an estimate of the power spectral density (PSD), and extract metrics from the IATS's power spectrum, that allows the detection of anomalous behaviour.

Formalizes a generic framework for modelling traces. We propose a method that allows the mapping of the information contained in an event trace to times series. The trace model presented in this paper is an improvement over previous work [41] with clearer definitions to decompose a trace into signals from which IATS can be extracted. The new trace model includes a more consistent and simplified notation and fixes minor inconsistencies present in [41]. We now focus on the concept of *base signal*, the simplest unit of information that can be associated with an event generator; then we extend this to the idea of aggregated signals, those that are obtained by joining base signals; meanwhile, the concept of *channel* is presented as a special case of an aggregated signal.

Improves metrics, and classification. Instead of using the Fourier Transform we now present IATS as random variables of real-time applications with stochastic execution times [25]. Therefore we shift the signal processing technique to nonparametric power spectral density estimation [29]. Based on the new signal processing approach, we define new metrics. For example, the DC significance is now computed from the PSD of the IATS. We also include multiple peaks in for the peak/frequency metric which we think represents a noticeable improvement compared to our previous work [41]. To extract multiple dominant peaks from the power spectrum of an IATS, we adapted an algorithm presented in [35] that determines a threshold above which a wide-sense stationary (WSS) signal shows dominant peaks. We also changed the classification method to Mahalanobis distance (MD) which improves our classifier performance by incorporating the correlation of the metrics when calculating distances.

Demonstrates the feasibility of using signal processing. We show that SiPTA outperforms other well-known methods to detect anomalies that change the power spectrum of a task IATS. In our experimental setup, we run SiPTA along with two anomaly detection methods (Markov Model, and Neural Network) on a set of comprehensive scenarios. The results show that the current implementation of SiPTA can detect all anomalous traces. We also discuss why our tool can perform better than the well-established methods used for performance comparison. The main reason SiPTA outperforms other approaches is the novel approach to trace modelling and the concept of extracting metrics from IATS power spectra.

In our study, we restrict our attention to traces collected from embedded devices. We evaluate SiPTA through the analysis of traces generated from QNX RTOS [2]. The systems targeted in this study are deployed on commercial platforms (i.e., a hexacopter, a car infotainment system, an a phone OS). The variety of targeted platforms cover a wide range of execution scenarios. We also chose to create own datasets, since the established datasets [1, 3] are unsuited for embedded systems and are under criticism [12, 27].

The remainder of the paper is organized as follows: Section 2 is a brief presentation of work related to anomaly detection; Section 3 presents an overview of the proposed methodology for detecting anomalies; Section 4 presents the trace model, a set of definitions and operations to extract times series from the trace; Section 5 introduces the frequency domain metrics computed from inter-arrival time series; Section 6 presents the raining and classification processes; Section 7 outlines the experimental setup and briefly discusses the alternative approaches toward we compare SiPTA's performance; Section 8 presents the results of our experimental setup which are then discussed in Section 9 along with an analysis on threats to validity; finally we give the conclusions in Section 10.

2 RELATED WORK

Detection of anomalies can be done: (i) online, i.e., *during runtime* or (ii) offline, i.e., *once the program has finished execution.* The offline approach considers an entire trace of a system execution scenario for analysis. The online method works on streams of execution events collected while the system runs; aiming to detect anomalies on-the-fly. Therefore, online anomaly detection techniques are suited for monitoring and active corrective measures. They usually do so by adapting the threshold of the anomaly score according to the captured stream. Offline detection techniques will, therefore, need some adaptation to be used at runtime, mainly in the amount of data required to detect anomalies. However, as pointed out in [10], some offline techniques cannot be used for online anomaly detection as they need to process the entire trace before deriving a conclusion.

In the context of detection of anomalies from sequential traces, most of the research effort has focused on the anomaly detection of operating system events. A recent survey paper [10] summarized the progress in that research area. They discussed two major approaches. The first uses Markovian modeling [38] to study the probabilistic characteristics of event transitions, extending from first-order to higher-order Markov Models and some equivalent methods as probabilistic suffix trees (PST), and sparse Markov trees (SMT). The second method models the event transition states through Finite State Automata (FSA) and Hidden Markov Models (HMM) methods. Besides these, there are other approaches to anomaly detection that do not fall into either of these two categories. These alternatives to anomaly detection exploit mathematical concepts that are widely used in signal processing. For instance, wavelet transform was used in [11, 15, 24, 30, 31] and Fourier transform was used in [42]. Their work focused on detecting anomalies in network-based systems. In contrast, our tool introduces the idea of using frequency characteristics of event transitions.

3 OVERVIEW

Our work targets timing anomalies by analyzing traces from systems with recurring periodic processes. Systems build around the concept of periodic tasks (or processes) are prevalent in the real-time embedded systems domain [22]. Our approach relies on the intuition that periodic task sets should produce pseudo-recurrent sequences of events. Therefore, timing anomaly detection can be accomplished by comparing time-stamps metrics computed from unknown traces to those computed from traces collected under normal operation. The goal is to find evidence to support the hypothesis that an unknown trace is generated by a known system. The alternative is to classify the trace as anomalous. Detecting timing anomalies becomes a systematic process that requires a framework allowing the modelling of traces, training the classifier, and classification.

A general work-flow of our framework is depicted in Figure 1; it is aimed at detecting anomalies using the offline approach. The tool is composed of three modules organized around two phases. The training phase requires a set of reference traces from which metrics are computed and analyzed to create the normal model. During the analysis phase, the tool computes test metrics, these metrics are compared to the normal model, and a classification score is given to the test trace. The classification algorithm computes the MD distance of test metrics to the normal model and assigns a classification score based on a classification threshold. Each block in Figure 1 performs the following operations.

Traces contain information describing the functioning of the system. The information in a trace is commonly composed of event names, time-stamps, and process names. SiPTA requires time-stamped entries.

The preprocessor extracts signals from traces, computes their time series, and feeds them into the underlying signal processing engine. Signals are subsets of the trace obtained by filtering it to entries for a specific type of event (e.g., entries for the *syscal=open, and PID=2*). There are two types of signals: a *base signal* contain entries for a specific event, while an *aggregated signal* is the union of two or more base signals. Time-series are computed from signals before entering the features extraction engine.

The features extraction engine takes time-series and outputs their frequency domain metrics. For the training traces, the set of its metrics represent a point in the multi-dimensional *normal behaviour space*. The set of metrics extracted from a trace is the *metrics tuple*.

The trainer takes a set of *metric tuples* computed from reference traces to create the *normal model* that will be used during the classification phase. The trainer can automatically identify the signals needed to compute the model. The model is a reduced set of metric tuples to which test traces will be compared during classification.

The binary classifier computes the distance of a *test metrics tuple* to the *normal model* by using the Mahalanobis distance. It determines whether a test trace is normal or anomalous by comparing the Mahalanobis distance to a set threshold, and gives a final normal/anomalous score.

4 TRACE MODEL

The trace model allows the disaggregation of traces into signals and time series. Times series are computed from signals. Therefore they contain the trace information in a suitable format for further signal processing. The following definitions provide the formalism to model traces so they can be precisely decomposed into signals and their corresponding time series.

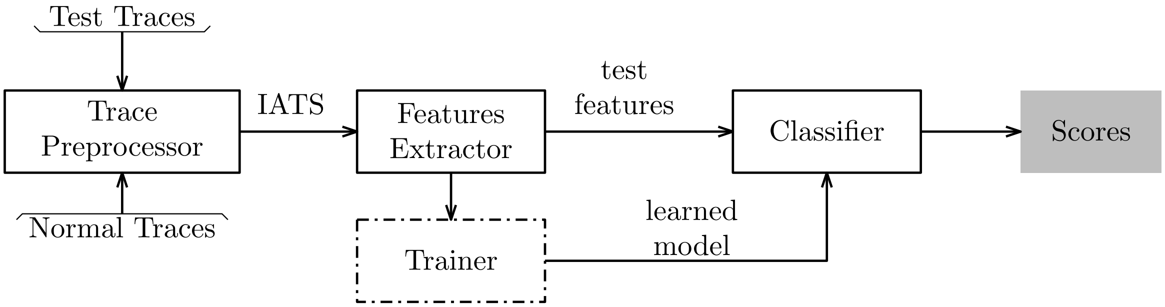


Fig. 1. Work-flow diagram for the offline implementation of SiPTA.

A trace is an array where each data point is a parameter value; rows are entries and columns event attributes. The parameter value $\rho_{i,j}$ is the value of the data point at location i, j, where i is the row index and j the column index. The rows in a trace are time-stamped entries, each recording a specific event. The columns match attribute names like time-stamps, process identifiers (e.g., `PID`) or the process state (e.g., `READY, BLOCKED`). The formalization is as follows:

Definition 4.1 (Trace). A trace $T \triangleq \langle e_1, ..., e_n \rangle$ is an ordered tuple of N entries, where e_i is the i^{th} row of T. The order is based on $e_i.t$; the timestamp value of entry e_i.

A trace of N entries is ordered if and only if $\forall e_i, e_j \in T$, the property $e_i.t \leqslant e_j.t$ holds, where entry indexes follow $i < j$. Note that we allow $e_i.t = e_j.t$ so the tracer can produce different entries with the same time-stamp. Duplicated entries are not allowed, thus if two or more trace entries have the same timestamp, their parameter sequence must be different. Therefore a trace is consistent if for any two entries e_i, e_j having $e_i.t = e_j.t$ the property $e_i.P_i \neq e_j.P_j$ holds.

Definition 4.2 (Trace Entry). A trace entry $e_i \triangleq \langle t_i, P_i \rangle$ is the i^{th} tuple of T. It consist of a time-stamp value t_i, and the M length parameter sequence P_i, where $M \triangleq |P_i|$. Each element in P_i is a parameter value $\rho_{i,j} : 1 \leqslant j \leqslant M$

Let P_i be the parameter sequence of entry e_i, the values in P_i determine a unique event occurred at time $e_i.t$, therefore each unique parameter sequence becomes an event identifier. The sequence of entries having the values of an event identifier is a base signal from which a sequence of timestamps can be extracted. The set of all event identifiers is the signal class set of the trace. Aggregated signals are obtained by filtering the trace to contain entries having parameter sequences that match one of many signal identifiers in the signal class. The formalization is as follows:

Definition 4.3 (Signal Class Set). The signal class set $\mathbb{S}$ of trace T defined as $\mathbb{S} \triangleq \mathrm{proj}(T(P))$ is the projection of T on the set of columns for parameter sequences.

We range over the elements of $\mathbb{S}$ using the notation $\mathbb{S}_i$, where $1 \leqslant i \leqslant |\mathbb{S}|$. Each element $\mathbb{S}_i \in \mathbb{S}$ defines a unique event identifier and it is used to extract signals from the trace.

Definition 4.4 (Base Signal). Signal $\mathcal{S}_i \triangleq T(\mathbb{S}_i) = \langle e_i, \ldots, e_k \rangle$ filters trace T to those entries for which $e_j.P = \mathbb{S}_i$.

Definition 4.5 (Aggregated Signal). Let $\mathbb{S}_i, \ldots, \mathbb{S}_k$ be elements of $\mathbb{S}$. The aggregated signal $\mathcal{S}_{i\cup\cdots\cup k} \triangleq T(\mathbb{S}_i, \ldots, \mathbb{S}_k) = \langle e_l, \ldots, e_n \rangle$ filters trace T to those entries for which $e_m.P = \mathbb{S}_i \vee e_m.P = \mathbb{S}_{[\cdot]} \vee e_m.P = \mathbb{S}_k$.

Since signals as defined in definitions 4.4 and 4.5 are tuples of entries they are not suitable for signal processing. The relevant information of a signal is contained in its time stamps. We first extract the real-time stamp sequence (RTSS) from a signal and then compute its IATS:

Definition 4.6 (Real-Timestamp Sequence). The L length real-timestamp sequence $r_i[l]$ of signal $\mathcal{S}_i$ is the projection of $\mathcal{S}_i$ on time-stamps, thus $r_i[l] \triangleq \mathrm{proj}(\mathcal{S}_i(t))$, and $L \triangleq |\mathcal{S}_i|$.

Real-timestamp sequences are monotonically increasing sequences, therefore not bounded, and hence not suitable for signal processing. Real-timestamp sequences are converted to inter-arrival time series as presented in [42].

Definition 4.7 (Inter-arrival time series IATS). Given an L real-timestamp sequence $r_i[l]$, its $N = L - 1$ inter-arrival time series $x_i[n]$ is the sequence after applying the first difference function to subsequent timestamps of $r_i[l]$:

$$x_i[n] \triangleq r_i[l] - r_i[l-1] : 1 \leqslant l \leqslant L - 1 \tag{1}$$

For example, periodic tasks generating at least one event per signal, IATS are bounded positive sequences. The maximum value an element can take being the task's period. Some tasks may not generate one event per signal within each job but their IATS will still be bounded by the maximum difference between two consecutive events. The following sections are based on the assumption that tasks will produce at least one event per base signal during the system's hyper-period.

4.1 Trace Example

Consider the trace shown in Table 1, it is a short view of an event trace recording time-stamps, process IDs, and system calls. According to our model, trace entry e_4 is the tuple $\langle 9, \langle 3, \texttt{read} \rangle \rangle$, the $syscal = $ "`read`" produced by the process with $PID = 3$ with time-stamp $t = 9$.

Let $\mathbb{S}_1 = \langle 3, \texttt{open} \rangle$ and $\mathbb{S}_2 = \langle 3, \texttt{read} \rangle$ be the first and second event identifiers in the signal class $\mathbb{S}$, their respective signals are:

$\mathcal{S}_1 = \langle e_1, \cdots, e_{22} \rangle$, and
$\mathcal{S}_2 = \langle e_2, e_4, e_6, e_7 \cdots, e_{23}, e_{24}, e_{26}, \cdots \rangle$, the aggregated signal
$\mathcal{S}_{1 \cup 2} = \langle e_1, e_2, e_4, e_6, e_7 \cdots, e_{22}, e_{23}, e_{24}, e_{26}, \cdots \rangle$ is the result of joining signals $\mathcal{S}_1$ and $\mathcal{S}_2$.

The real time-stamp sequence of signal $\mathcal{S}_{1 \cup 2}$ is $r_{1 \cup 2}[n] = \{0, 3, 9, 10, 13, \cdots, 33, 36, 42, 43, \cdots\}$ with IATS $x_{1 \cup 2}[n] = \{3, 6, 1, 3, \cdots, 3, 6, 1, \cdots\}$. In this example, the IATS shows a recurrent pattern of inter-arrival times: $(3, 6, 1)$.

Table 1. A simple trace with two element parameter sequences. The entry index number is added for reference.

Index	*Time*	*PID*	*Syscall*
1	0	3	`open`
2	3	3	`read`
3	3	4	`open`
4	9	3	`read`
5	10	4	`read`
6	10	3	`read`
7	13	3	`read`
8	14	4	`close`
...	...	...	...
20	30	4	`close`
21	32	3	`close`
22	33	3	`open`
23	36	3	`read`
24	42	3	`read`
25	42	4	`open`
26	43	3	`read`
27	45	4	`read`
...	...	...	...

5 FREQUENCY DOMAIN METRICS

The IATS extracted from real traces should be considered realizations of random processes. We motivate our work on the assumption that these processes are WSS when the time-stamps are produced by periodic tasks. Indeed, periodic tasks are known to be WSS in response times [13], and we loosely extend this idea to IATS based on the concept of the system's hyper-period [23, 33].

Consider a periodic task set Γ_N of N tasks, each task τ_i having period T_i. Under the worst-case scenario, the schedule of such task set will repeat after the hyper-period $T_H = \mathrm{lcm}(T_1, \ldots, T_N)$ [23, 33]. From the IATS perspective, this means that N_E the number of events of a signal is constant within one hyper-period, while the amplitude of inter-arrival times is determined by the schedule, hence all the IATS Γ_N generates are periodic. Therefore, for any signal $\mathcal{S}_i$, with N_{iE} events per hyper-period, we have $x_i[n] = x_i[n + N_{iE}]$, and $\sum_{n=0}^{N_{iE}-1} x_i[n] = T_H$ over any hyper-period. Relaxing the worst-case execution condition while keeping N_{iE} constant allows the amplitude of inter-arrival times in $x_i[n]$ to vary from one hyper-period to another while keeping the sum constant over T_H. However, in the general case, the IATS generated by Γ_N are not periodic $(x_i[n] \neq x_i[n + N_{iE}])$ but we conjecture they are WSS.

The expected value and variance of the random IATS generated by Γ_N will be time invariant. For the random variable X_i, the ensemble of all possible realizations of the random process, and having N_{iE} events per hyper-period the expected value and variance of X_i are:

$$\mathrm{E}[X_i] \triangleq \mu_i = \frac{\sum_{n=0}^{N-1} X_i}{kN_{iE}} = \frac{kT_H}{kN_{iE}} = \frac{T_H}{N_{iE}} \tag{2}$$

$$\mathrm{var}[X_i]] \triangleq \sigma_i^2 = \mathrm{E}[(X_i - \mu_i)] = \mathrm{E}[X^2] - \mu_i^2 \tag{3}$$

where $k \geqslant 1 : k \in \mathbb{R}$ is the number of hyper-periods covered by X_i. Indeed Equations (2), and (3) shows that IATS are mean and variance ergodic, being this the first condition for wide-sense stationarity [26]. Proving that IATS are indeed WSS is beyond the scope of this work. Nevertheless, we assume that IATS generated by a periodic task set are realizations of WSS processes. We support our intuition on [13], and the assumption that the implementation of the system task's code remains unchanged during trace collection. Thus, we compute the PSD of IATS and extract suitable frequency domain (FD) metrics that are used for timing anomaly detection.

5.1 Overview of PSD Estimation.

The recurrent nature of an event generator can be better examined throughout the power spectral density (PSD) of its IATS. The PSD $S(\omega)$ is a continuous function of frequency a discrete representation is usually computed using some PSD estimation method. An example of two realizations of IATS generated by a periodic task is shown in Figure 2. Note that even though the IATS have different time domain representations they have almost identical power spectra. The peaks of the Welch estimate $\hat{W}[k]$ shown in Figure 2 b,d are those related to the pseudo-periodic sequence of inter-arrival times of its underlying generator. Conversely, the power spectrum of an anomalous IATS will be different to that of its normal counterparts; this is true, even if the anomalous generator remains WSS. The reason for the difference being that an anomalous generator will have different moment statistics when compared to a normal generator. Therefore, normal and anomalous IATS will have different PSD that enables binary classification.

There are two main approaches to PSD estimation: parametric, and non parametric [26, 28]. Parametric methods work under the assumption that the generator (tasks) of the processed time series (IATS) follows some spectral density function whose parameters can be estimated from a realization of the signal. nonparametric PSD methods are used when no assumptions can be made

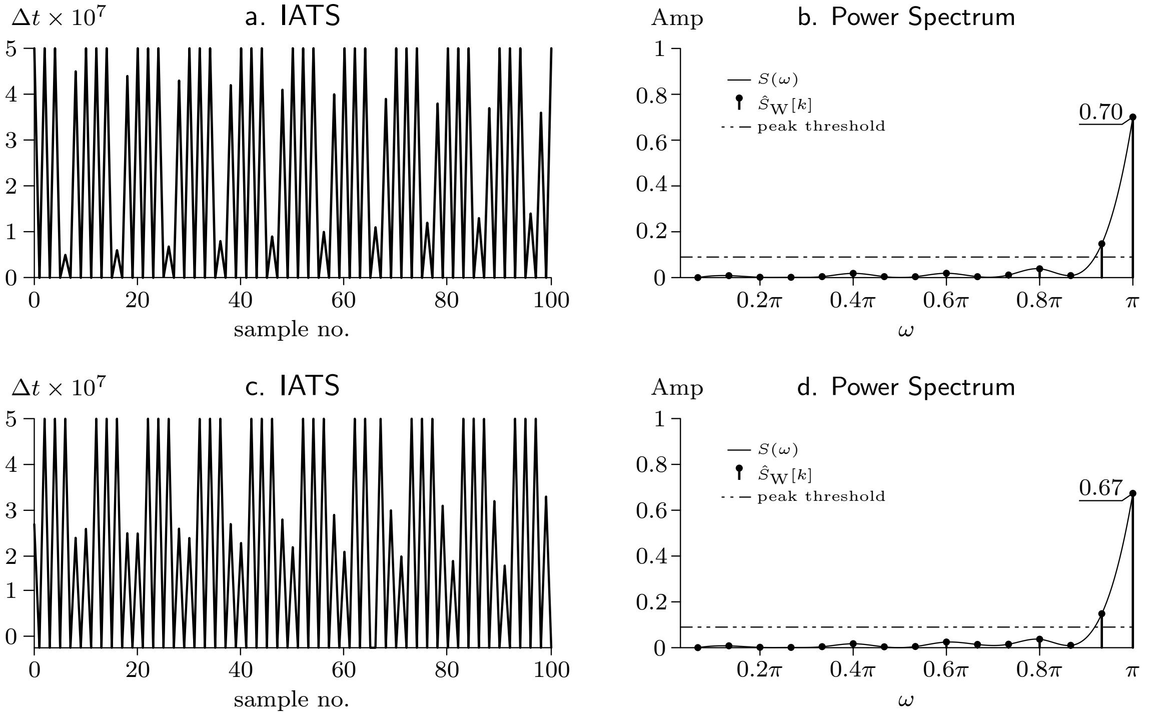


Fig. 2. Two segments of 100 samples of IATS from the same generator in a, and c with their corresponding normalized true PSD ($S(\omega)$), and Welch peridogram ($\hat{W}[k]$) on the right. Note how although the IATS segments show differences, their power spectra are almost identical.

about the properties of the generator. In our case, we cannot assert the existence of a model for generators and therefore we are left to apply nonparametric PSD.

Among the different nonparametric methods for PSD estimation, the periodogram by Schuster [32] is the most basic and best known. Although a powerful tool, the periodogram is not a consistent estimator of the power spectral density function $S_i(\omega)$ [21]. Averaging periodograms is a solution that improves the PSD estimate by reducing the periodogram variance. First, the sequence $x_i[n]$ is split into a set of smaller segments, then the periodogram of each segment is computed, and finally, the average periodogram is computed from the set of *segment periodograms*. Two classic methods use averaging techniques: Bartlett [7, 8] divides the sequence into a set of contiguous nonoverlapping segments. The method proposed by Welch [36, 37] finds a set of modified periodograms by windowing a set of overlapping segments and then averaging the modified periodograms.

Averaging periodograms improves the consistency of the estimated PSD by reducing the variance of the estimate at the expense of resolution in the frequency domain. We chose the Welch method for two reasons (i) it estimates the PSD using more segments, and therefore the variance converges more quickly, and (ii) individual periodograms are computed from windowed segments that reduce spectral leakage. The method to compute the Welch periodogram is as follows:

Let $x_i[n]$ be the IATS of signal $\mathcal{S}_i$. First, split $x_i[n]$ into a set of L overlapping segments of length M. Then compute the segment modified periodogram:

$$\tilde{P}_i^{(l)}[k] = \frac{1}{MU}\left|\sum_{n=0}^{M-1} x_i^{(l)}[n]w[n]e^{-\mathrm{j}\frac{2\pi n}{M}}\right|^2, l = 0, 1, \ldots, L-1 \tag{4}$$

Where U is the the normalization factor for the window $w[n]$:

$$U = \frac{1}{M} \sum_{m=0}^{M-1} w^2[m] \tag{5}$$

The Welch PSD estimate is found by averaging the modified periodograms computed using (4):

$$\hat{S}_{\mathrm{W}i}[k] = \frac{1}{L} \sum_{l=0}^{L-1} \hat{P}_i^{(l)}[k] \tag{6}$$

The Welch periodogram is even and centered at $\omega = 0$, where $-\pi < \omega \leqslant \pi$. Therefore only positive frequencies need to be considered.

5.2 DC significance metric

Consider a direct current (DC) IATS of amplitude A. This is in general, an uninteresting sequence, but in our case, it means that the generator is fully periodic because it generates an event every A units of time. Since, in the FD, all the energy is concentrated at the DC component the ratio of energy at $\omega = 0$ to the total signal energy can be used as an indirect measure of how periodic is the generator (not the IATS). In practice, we allow some variation around the ideal amplitude, i.e., $A - \epsilon \leqslant x_i[n] \leqslant A + \epsilon : \epsilon \ll A$. Under this conditions, $S_i(\omega) \neq 0$ for $|\omega| > 0$. However, since $\epsilon \ll A$, most of the signal energy will still be concentrated at $\omega = 0$, while the PSD will be almost zero for any non-zero frequency. Hence, we can use the ratio of energy at $\omega = 0$ to the total signal energy (see Equation (7)) as an indirect measure of an event generator's periodicity.

Definition 5.1 (DC significance metric). Let $\hat{S}_{\mathrm{W}i}[k]$ be the PSD estimate of IATS $x_i[n]$ computed by the Welch method. The DC significance $(\mathcal{D}_i)$ of the IATS $x_i[n]$ is:

$$\mathcal{D}_i \triangleq \frac{\hat{S}_{\mathrm{W}i}[0]}{\sum\limits_{k=0}^{N-1} \hat{S}_{\mathrm{W}i}[k]} \tag{7}$$

where $\hat{S}_{\mathrm{W}i}[0]$ is the DC component of $\hat{S}_{\mathrm{W}i}[k]$; i.e $S(\omega)|_{\omega=0}$.

When the IATS is a periodic sequence, the PSD of $x_i[n]$ will have significant harmonics at some $\omega \neq 0$, which decreases the DCs value; this means that the DCs is a good estimator of normal behaviour for DC generators or those whose $\mathcal{D}_i \approxeq 1$. However, most IATS will be far from being DC or even periodic sequences. Therefore a complementary metric is presented below.

5.3 Multi-peak frequency metric

When the IATS of a generator is a periodic sequence, dominant peaks will show at the fundamental frequency of the sequence period. The problem is that most real IATS are random and therefore noise will be present in their PSD. Among the undesirable features emerging in an IATS's PSD, low-frequency peaks are of particular interest because they are not related to the real periodicities of the underlying generator.

Let $x_i[n]$ be an IATS covering some number of hyper-periods, also let N be large and k small. The lowest frequency components of $\hat{S}_{\mathrm{W}i}[k]$ will be located at frequencies near zero, i.e., $\omega_{\pm k} = \pm\frac{2\pi k}{N}\Big|_{k \ll N} \approx \pm 0$. Since $\hat{S}_{\mathrm{W}i}[k]$ is even symmetric, our reasoning follows for $\omega \geqslant 0$. Recall that the expected recurrent sequence of an IATS should repeat with each hyper-period. Therefore, any dominant peak found in the IATS's PSD should be related to the sequence period. However, since the IATS is assumed to be WSS low-frequency dominant peaks may appear due to noise and spectral leakage. The problem is that a low dominant peak present at a frequency lower than the natural frequency of the recurrent sequence is not related to the normal behaviour of the system. To further complicate this problem, there is no guarantee that these peaks will always be present in the PSD at the same frequencies and amplitude. We actively remove low-frequency dominant peaks under the assumption that any peak present under some cutoff frequency ω_c can be attributed to noise present in $x_i[n]$ and not to the real recurrences in the IATS. As a result, we eliminate these undesirable peaks by filtering $x_i[n]$ using a high pass filter with cutoff frequency ω_c.

Consider $\hat{S}_{\mathrm{W}i}[k]$, the discrete PSD of filtered $x_i[n]$, $\hat{S}_{\mathrm{W}i}[k]$ should only contain the dominant FD peaks of $x_i[n]$ related to recurrent sequences of events. Moreover, a few number of peaks from $\hat{W}[k]$ will be significant. The multi-peak frequency metric $\mathcal{P}_i(\mathcal{S}_i, N_P)$ of IATS $x_i[n]$ is the metric containing the highest peaks of $\hat{S}_{\mathrm{W}i}[k]$, and is defined as follows:

Definition 5.2 (Multi-peak frequency metric). Let $\hat{S}_{\mathrm{W}i}[k]$ be the discrete PSD estimate of IATS $x_i[n]$, extracted from signal $\mathcal{S}_i$ and passed through a high pass filter with $\omega_{cut} \geqslant \omega_c$. Assume there are a minimum of N_P dominant peaks in $\hat{S}_{\mathrm{W}i}[k]$. The multi-peak frequency metric $\mathcal{P}_i(\mathcal{S}_i, N_P)$ of IATS $x_i[n]$ is the set of *amplitude, frequency* tuples in decreasing order of amplitude, and frequency.

$$\mathcal{P}_i(\mathcal{S}_i, N_P) \triangleq \left\{ \left\langle \hat{W}_i[k]_{N_P}, \omega_{k_{N_P}} \right\rangle, \ldots, \left\langle \hat{S}_{\mathrm{W}i}[k]_1, \omega_{k_1} \right\rangle \right\} \tag{8}$$

where $\hat{S}_{\mathrm{W}i}[k]_m \geqslant \hat{S}_{\mathrm{W}i}[k]_l$, and $\omega_{k_m} > \omega_{k_l}$ for any two peaks $\hat{S}_{\mathrm{W}i}[k]_m = \hat{S}_{\mathrm{W}i}[k]_l$.

Note that the order is given based on the amplitude first, then the frequency of the peaks for peaks of equal amplitude. For example, two equal peaks $\hat{S}_{\mathrm{W}i}[k]_m = \hat{S}_{\mathrm{W}i}[k]_l$, are ordered according to their frequencies: $\omega_{k_m} > \omega_{k_l}$. Based on the definition, there is no need to extract all the significant peaks from $\hat{S}_{\mathrm{W}i}[k]$, but we need a method from which we can determine N_P which it is introduced in the next section.

6 TRAINING AND CLASSIFICATION

Since we assume that a periodic task is a random process, we should expect some variance in the power spectrum of its IATS realizations. To compensate for this variance, we train a normal model from a set of reference (normal) traces. A reference trace is one with a high likelihood of containing expected sequences of inter-arrival times. Normal traces are recorded under known operating conditions where no anomalous activity is suspected. For example, a trace from a car that passes industry standard tests is likely to be normal and therefore can be labelled as such. Our reference model will be built from information extracted from a training set $\mathbb{T}^{\mathfrak{t}} \triangleq \{T^{\mathfrak{t}1}, \ldots, T^{\mathfrak{t}\mathfrak{n}_{\mathfrak{t}}}\}$ composed of $\mathfrak{n}_{\mathfrak{t}}$ *reference* traces.

Analysis traces are recorded under any operating condition, and no assumptions are made about its conformity to normality. For example, a trace recorded in a normal driving scenario may be considered an analysis trace. A set of analysis traces $\mathbb{T}^{\mathfrak{a}} \triangleq \{T^{\mathfrak{a}1}, \ldots, T^{\mathfrak{a}\mathfrak{n}_{\mathfrak{a}}}\}$ is composed of $\mathfrak{n}_{\mathfrak{a}}$ traces whose classification is unknown. We consider each trace $T^{\mathfrak{a}i}$ of $\mathbb{T}^{\mathfrak{a}}$ independently and assign a classification score to each element of $\mathbb{T}^{\mathfrak{a}}$.

The classifier uses the reference model as comparison standard during classification of analysis traces. The reference model is constructed in two steps: First, each trace in $\mathbb{T}^{\mathfrak{t}}$ is checked for signal consistency, which requires the verification of signal existence and signal length. Second, a model containing the metrics for the signals involved in classification is computed from the traces in $\mathbb{T}^{\mathfrak{t}}$.

6.1 Signal consistency

The first step toward a reference model is the determination of what signals are present in all traces of $\mathbb{T}^{\mathfrak{t}}$. Classification based on Mahalanobis distance requires a model composed of a fixed number of independent variables. We consider each element in a metric an independent variable. Therefore, all traces in $\mathbb{T}^{\mathfrak{t}}$ must contain the same signals. Recall those signal identifiers are elements of the trace's signals class set (see Definition 4.3) used to extract signals from the trace (see Definitions 4.4, and 4.5). Hence, we define a training signal class set for $\mathbb{T}^{\mathfrak{t}}$ as:

Definition 6.1 (Training signal class set). Let $\mathbb{S}^{\mathfrak{t}i}$ be the signal class set of $T^{\mathfrak{t}i}$, the i^{th} trace in $\mathbb{T}^{\mathfrak{t}}$. The training signal class set, $\mathbb{S}^{\mathfrak{t}}$ of $\mathbb{T}^{\mathfrak{t}}$ defined as $\mathbb{S}^{\mathfrak{t}} \triangleq \mathbb{S}^{\mathfrak{t}1} \cap \mathbb{S}^{\mathfrak{t}2} \cap \ldots \mathbb{S}^{\mathfrak{t}\mathfrak{n}_{\mathfrak{t}}}$ is the subset containing the signals common to all $T^{\mathfrak{t}i} \in \mathbb{T}^{\mathfrak{t}} : 1 \leqslant i \leqslant \mathfrak{n}_{\mathfrak{t}}$.

Note that $\mathbb{S}^{\mathfrak{t}} \subseteq \mathbb{S}^{\mathfrak{t}i}, \forall T^{\mathfrak{t}i} \in \mathbb{T}^{\mathfrak{t}}$, hence non shared signals are excluded from the training signal class set. There are various reasons to have excluded signals without affecting the model. For example, a normal trace may contain entries for initialization commands that might not be present in other normal traces. Moreover, if the signal class set $\mathbb{S}^{\mathfrak{a}i}$ of trace $\mathbb{T}^{\mathfrak{a}i}$ does not contain $\mathbb{S}^{\mathfrak{t}}$, i.e., $\mathbb{S}^{\mathfrak{a}i} \cap \mathbb{S}^{\mathfrak{t}} \neq \mathbb{S}^{\mathfrak{t}}$, there are missing signals in $\mathbb{T}^{\mathfrak{a}i}$, and the trace can be classified as anomalous without further analysis.

A comprehensive model for normal behaviour can be built by using the elements of $\mathbb{S}^{\mathfrak{t}}$ as signal identifiers. However, signals must contain enough information to enable the computation of PSD estimates by the Welch method. If signals are too short, $\mathbb{S}^{\mathfrak{t}}$ can be further reduced. Some signals may be short without compromising system's analysis. For example, sporadic tasks that are rarely activated, or events generated when a task occasionally takes some execution path will be inevitably short when transformed into IATS. The minimum length $L_{min}(\mathcal{S}_i)$ of signal $\mathcal{S}_i$ depends on the desired frequency domain resolution when computing $\hat{S}_{\mathrm{W}i}[k]$, and the number of segments required to estimate the PSD.

Definition 6.2 (Short signal). Let N_i be the length of signal $\mathcal{S}_i$, and $L_{min}(\mathcal{S}_i) \triangleq K_i M_i + 1$ the minimum length required to compute an accurate PSD estimate of $x_i[n]$. Where K_i is the number of non overlapping segments, and M_i the segment length required for an accurate computation of $\hat{S}_{\mathrm{W}i}[k]$. Signal $\mathcal{S}_i$ is short if $N_i < L_i(\mathcal{S}_i)$.

Finding appropriate values for K_i, and M_i is hard without knowing task periods in a task set and the expected number of events per hyper-period. The difficulty to accurately determine the minimum length of an IATS is one reason we limited our study to offline anomaly detection. Having large traces covering a substantial amount of time and containing numerous events reduces the chances that not enough information is present in the trace.

Short signals cannot be included in the normal model because computing accurate metrics is not possible. Also, note that $\mathbb{S}^{\mathfrak{t}}$ contain signal identifiers only for base signals. In most cases, not all base signals need to be included, while aggregated signals may better explain the behaviour of a task. Nevertheless, $\mathbb{S}^{\mathfrak{t}}$ is the foundation to construct such set of signals.

It is hard to determine what signal identifiers better represent the behaviour of the system, or agree on a method to define rules from which aggregated signals can be defined by joining base signals whose identifiers are taken from $\mathbb{S}^{\mathfrak{t}}$. The main goal should be the minimization of the number of signals to form from the elements of $\mathbb{S}^{\mathfrak{t}}$. A simple but effective rule we found useful is the idea of a channel. A channel is the aggregated signal resulting from joining all the base signals associated with an even generator whose identifiers belong to $\mathbb{S}^{\mathfrak{t}}$.

Definition 6.3 (Channel). Let $G = \langle l, \ldots, n \rangle$ be the set of column indexes associated to event attributes that uniquely identify event generators in a trace. Also let $\mathbb{G}^{\mathfrak{t}} \triangleq \mathrm{proj}(\mathbb{S}^{\mathfrak{t}}(G))$, be the projection of the training class set on G, and $P_i(\mathbb{G}^{\mathfrak{t}}_i)$ a logic function that returns true if $P_i.G = \mathbb{G}^{\mathfrak{t}}_i$, where $P_i.G$ is the subset of parameter values $\rho_{i,m} : m \in G$. The channel $\mathcal{C}_i \triangleq T(\mathbb{G}^{\mathfrak{t}}_i)$ filters trace T to those entries for which $e_i.P(\mathbb{G}^{\mathfrak{t}}_i)$ holds.

6.2 Reference model

The reference model is an ordered collection of metrics computed from signals formed from the elements of $\mathbb{S}^{\mathfrak{t}}$. Each trace will provide a tuple of metrics per analyzed signal. The metrics of a trace can be interpreted as a normal point in the multidimensional space of normal behaviour.

Definition 6.4 (Metrics Tuple). Let T be a trace either in the training or analysis set. And let $\mathbb{G}^{\mathfrak{t}}$ be the set from which channels are extracted from T. The metrics tuple $\mathcal{M}$ of T is:

$$\mathcal{M} \triangleq \langle \mathcal{D}(\mathcal{C}_1), \mathcal{P}(\mathcal{C}_1, N_{P1}) \rangle, \ldots, \langle \mathcal{D}(\mathcal{C}_L), \mathcal{P}(\mathcal{C}_L, N_{PL}) \rangle \tag{9}$$

where $L = |\mathbb{G}^{\mathfrak{t}}|$, and N_{Pi} is the number of peaks to extract from channel $\mathcal{C}_i$.

The reference model of $\mathbb{T}^{\mathfrak{t}}$ can now be defined in terms of the metrics tuple of each trace, arranged in matrix form.

Definition 6.5 (Reference model). Let $\mathbb{T}^{\mathfrak{t}}$ be a training set of $\mathfrak{nt}$ traces, with signal set $\mathbb{M}^{\mathfrak{t}}$. The reference model $\mathcal{R}^{\mathfrak{t}}$ of $\mathbb{T}^{\mathfrak{t}}$ is:

$$\mathcal{R}^{\mathfrak{t}} \triangleq \begin{matrix} \mathcal{D}_1(\mathcal{S}^{\mathfrak{m}1}) & \mathcal{P}_1(\mathcal{S}^{\mathfrak{m}1}) & \cdots & \mathcal{D}_1(\mathcal{S}^{\mathfrak{m}L}) & \mathcal{P}_1(\mathcal{S}^{\mathfrak{m}L}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{D}_{\mathfrak{nt}}(\mathcal{S}^{\mathfrak{m}1}) & \mathcal{P}_{\mathfrak{nt}}(\mathcal{S}^{\mathfrak{m}1}) & \cdots & \mathcal{D}_{\mathfrak{nt}}(\mathcal{S}^{\mathfrak{m}L}) & \mathcal{P}_{\mathfrak{nt}}(\mathcal{S}^{\mathfrak{m}L}) \end{matrix}$$

Each row in $\mathcal{R}^{\mathfrak{t}}$ contains the elements of the metrics tuple for a trace in the training set. Therefore, the reference model contain the information required to establish a comparison between the set of training traces to an unknown trace. Each column in $\mathcal{R}^{\mathfrak{t}}$ is an independent variable in the normal behaviour of the system. Meanwhile, each row is a point of the multidimensional, normal space. An accurate normal operation mode for a system will contain a set of points that cluster in some region of the multidimensional, normal space.

6.3 Peak detection

The automatic detection of dominant peaks is achieved by applying algorithm 1, an adaptation of the `PERIOD_HINTS` algorithm. Originally proposed by Vlachos et al. in [35]; `PERIOD_HINTS` determines a threshold above which any component in the power spectrum becomes a period hint. A period hint is a dominant FD peak that may be associated with a real periodicity of the stationary signal. `PERIOD_HINTS` is based on the periodogram and also contain some features not needed in SiPTA. For example, in [35] some hints may be discarded because they are likely to represent unreal periodicities in the targeted times series, since our requirements are different we adapted the algorithm.

Our version, called `MULTI_PEAKS` and shown in Algorithm 1 is a modified version of `PERIOD_HINTS` that uses filtered time series and the Welch method to compute significant FD peaks. Recall that spectral leakage is a real problem that arises when computing periodograms, the Welch method reduces leakage by computing modified periodograms for each segment. Another problem that contributes to leakage is the segment length, which we keep constant for each signal. Even when the same signal has IATS of different length for different normal traces; the peaks due to leakage will appear around the same frequencies and will have about the same amplitude. As mentioned before, low-frequency peaks are removed by filtering the IATS with a high pass filter. `MULTI_PEAKS` returns N_P the number of FD peaks found, and t_h the threshold above which FD peaks become dominant.

Recall that we need to know the number of peaks to retrieve from a power spectrum when extracting the MPFm, but nothing is assumed about the number of dominant peaks in a signal's PSD. Since dominant peaks should be present in all normal traces, with some variance allowed, we can

Algorithm 1 MULTI_PEAKS

```
Input: $x[n], h[n], M$
Output: $N_P, t_h$ {Number of peaks and threshold of significant peaks}
  $P_{max} = \emptyset$ {vector of maximum amplitudes}
  $N_P = 0$
  $xf[n] = \texttt{filter}(x[n], h[n])$

  for $l = 1$ to 100 do
    $x_l[n] = \texttt{permute}(xf[n])$
    $\hat{W}_l[k] = \texttt{welch}(x_l[n], M)$ {segment length: $M$, overlap: $50\%$}
    $\texttt{insert}(P_{max}, \texttt{max}(\hat{W}_l[k]))$
  end for

  $\texttt{sort}(P_{max}, \texttt{descending})$
  $t_h = P_{max}[10]$ {$10^{th}$ element is $90^{th}$ largest power}

  $\hat{W}[k] = \texttt{welch}(xf[n], M)$
  $\hat{W}_{sort}[k] = \texttt{sort}(\hat{W}[k], \texttt{descending})$

  while $\hat{W}_{sort}[N_P] \geqslant t_h$ do
    $\texttt{increment}(N_P)$
  end while

  return $N_P, t_h$
```

use MULTI_PEAKS to mine that information from the training set. We propose MODEL_PEAKS, shown in Algorithm 2, given a channel identifier, it computes the thresholds and peaks for each trace in the training set, then finds the average threshold and the average number of peaks. MODEL_PEAKS checks that all channels have at least the average number of peaks before returning the average threshold and the number of peaks (rounded down to the nearest integer). With the information returned from MODEL_PEAKS we can proceed to the extraction of the MPFm as defined in Definition 5.2.

6.4 Classification

Based on definitions 6.1, and 6.2, test traces can be deemed anomalous when the signal class of the analysis trace does not contain the same signals in the training signal class. Formally we say that some analysis trace T^{ai} is anomalous if $\mathbb{S}^{ai} \cap \mathbb{S}^{t} \neq \mathbb{S}^{t}$, because a signal to form a channel is missing in T^{ai}. A similar reasoning applies to short signals because there are not enough recorded events to compute an accurate PSD. These two conditions help in the classification of anomalous traces that do not contain the type or amount of information expected in a normal trace.

Since we consider only one normal space, the classification can be accomplished by computing the distance of an unknown point to the centroid of the normal cluster. The Mahalanobis distance is our choice because it takes into account the covariance of the independent variables.

Definition 6.6. Let $\mathcal{M}^{ai}$ be the i^{th} metrics tuple of analysis set $\mathbb{T}^{a}$. The distance of $\mathcal{M}^{ai}$ to the reference model $\mathcal{R}^{ai}$, extracted from training set $\mathbb{T}^{t}$ is computed by the squared Mahalanobis Distance:

Algorithm 2 MODEL_PEAKS

```
Input: $\mathbb{G}_i^{\mathfrak{t}}, \mathbb{T}^{\mathfrak{t}}, h[n], M$
Output: $\bar{N}_{Pi}, \bar{t}_{hi}$ {Average number of peaks, and threshold for channels $\mathcal{C}_i^{\mathfrak{t}}$}
  $Thres = \emptyset, N\_Pk = \emptyset$

  for $k = 1$ to $\mathfrak{n}_{\mathfrak{t}}$ do
    $[t_{hi}^{\mathfrak{t}k}, N_{Pi}^{\mathfrak{t}k}] = \texttt{MultiPeaks}(\mathcal{C}_i^{\mathfrak{t}k}, h[n], M)$
    $\texttt{insert}(Thres, t_{hi}^{\mathfrak{t}k})$
    $\texttt{insert}(N\_Pk, N_{Pi}^{\mathfrak{t}k})$
  end for

  $\bar{t}_{hi} = \texttt{Average}(Thres)$
  $\bar{N}_{Pi} = \texttt{Average}(N\_Pk)$

  for $k = 1$ to $\mathfrak{n}_{\mathfrak{t}}$ do
    if $\texttt{CountPeaks}(\mathcal{C}_i^{\mathfrak{t}k}, \bar{t}_{hi}) < \lfloor \bar{N}_{Pi} \rfloor$ then
      return FAIL, $k$
    end if
  end for

  return $\lfloor \bar{N}_{Pi} \rfloor, \bar{t}_{hi}$
```

$$D_M^2(\mathcal{M}^{\mathfrak{ai}}, \mathcal{R}^{\mathfrak{t}}) = \left(\mathcal{M}^{\mathfrak{ai}} - \hat{\mathcal{R}}^{\mathfrak{t}}\right)^T \Sigma^{-1} \left(\mathcal{M}^{\mathfrak{ai}} - \hat{\mathcal{R}}^{\mathfrak{t}}\right) \tag{10}$$

Where $\hat{\mathcal{R}}^{\mathfrak{t}}$ is the columns mean vector of the model $\mathcal{R}^{\mathfrak{t}}$, and Σ^{-1} is inverse of the covariance matrix of $\mathcal{R}^{\mathfrak{t}}$.

If the traces in $\mathbb{T}^{\mathfrak{t}}$ are similar to each other, their pairwise MD will be small. Therefore, any normal trace should also be close to the reference model and $D_M^2(\mathcal{M}^{\mathfrak{ai}}, \mathcal{R}^{\mathfrak{t}})$ should also be small. The more different an analysis trace is to those in the training set, the larger the MD. The classification is achieved by setting a classification threshold θ_{M} and a classification function as follows:

$$S(D_M^2(\mathcal{M}^{\mathfrak{ai}}, \mathcal{R}^{\mathfrak{t}}), \theta_{\mathrm{M}}) = \begin{cases} 1, & D_M^2(\mathcal{M}^{\mathfrak{ai}}, \mathcal{R}^{\mathfrak{t}}) \leqslant \theta_{\mathrm{M}} \\ 0 & otherwise \end{cases} \tag{11}$$

where a score of one means the trace is normal, and anomalous otherwise.

Proper determination of a good choice for θ_{M} will require a performance analysis for each specific application. A well established method to conduct such performance analysis is receiver operational characteristic. We perform ROC analysis on SiPTA for a set of execution scenarios in the following section.

7 EXPERIMENTAL EVALUATION

While SiPTA in theory is well suited to the problem domain of periodic systems, we followed up our theory work with an experimental evaluation to provide quantitative results. This section describes the experiments we conducted to test our tool, evaluate its performance, and discuss the results.

7.1 Experimental Setup and Workload

We ran our experiments on the QNX Neutrino 6.4 real-time operating system. While the framework is independent of specific system traces, we had access to three QNX based embedded platforms, from which we can gather traces: a hexacopter [4], a QNX CAR infotainment unit, and Blackberry phones running QNX. This allows us to gather traces from very different execution contexts to properly evaluate SiPTA.

Data was collected by the QNX kernel logging facility, which provides tracing capabilities through QNX tracelogger. A trace collected from tracelogger contain a chronological order of system events such as system calls, message passes, interrupts, I/O, etc. Each trace entry corresponds to a system event and complementary information, such as the source/destination of a message pass. A sample trace is shown in the following snippet:

```
TIMESTAMP, CPU, EVENT,      PID, PROC, Details
t1,        1,   PROCCREATE, 1,   A,    PPID: 0 ...
t2,        1,   THCREATE,   1,   A,    ...
t3,        2,   INT_ENTR,   1,   A,    ...
t4,        2,   INT_EXIT,   1,   A,    ...
t5,        1,   MSG_SND,    1,   A,    To: B ...
t6,        1,   MSG_RECV,   3,   B,    From: A ...
.....
tn,        1,   KER_CALL,   1,   A,    SIGKILL ...
```

The above snippet shows the life-cycle of a process as recorded by the system. The process A, once created, spawns a thread, services an interrupt and sends a message to some other process B before being terminated.

Our experimental setup requires a subset of all the information contained in a trace. Specifically, we only consider the following event attributes: *class, event, time, pid, and process name*. This is consistent with related work [9] which reduces the used attributes to similar lists.

The experimental workload consists of six different execution scenarios for the four systems under test. Each scenario ran between 10 to 20 seconds, and the traces contain all logging information. Within one scenario each trace covers about the same runtime. For any given scenario, traces are arbitrarily assigned to the training $\mathbb{T}^{t}$, or analysis set $\mathbb{T}^{a}$. All traces are labelled as normal or anomalous so we can check classification performance. Depending on the detection technique, we may need only normal traces during training (SiPTA, Markov model) or a combination of normal and anomalous traces (Neural Networks). The analysis set can contain a set of normal traces, a set of anomalous traces, or a combination of both.

Hexacopter: The hexacopter system is an unmanned aerial vehicle (UAV) platform, which implements non-trivial software and systems control [4]. The platform runs on beaglebone white with ARM Cortex-A8 processor and comprise a networked system of hardware and software components. The hexacopter has been field tested through several mission-critical applications including: iceberg monitoring on the open sea and creating infrared maps over critical Canadian Solar infrastructure. For our experiments, we created two test scenarios. In both scenarios, the normal traces correspond to the hexacopter running normally:

Scenario	Anomalies
Run normally (I)	Periodic recursive listing
Run normally (II)	Run tight loop process

QNX CAR infotainment device: The second set of scenarios uses QNX CAR [5], which is the leading in-car infotainment system. The device was running QNX Neutrino on an *i.MX6Q*

(Sabre lite) board with an ARM Cortex A9 quad-core processor. We created the following scenarios on this platform:

Scenario	Anomalies
Idle	Induced network traffic, user inputs
Play MP3	Seeking, fast-forward, different song
Play Video	Seeking, fast-forward, different video
Run *top* command	Induced network traffic, more shell tasks
Variable speed ping	Change ping rate from five to two seconds

QNX-based BlackBerry Z10 phone: The third set of scenarios uses the BlackBerry phone. These phones run a modified version of QNX Neutrino, however, the tracelogging facilities are still available. We created the following set of scenarios on this platform:

Normal trace	Anomalies
Record video	Zooming and toggling the camera, light on/off
Play youtube video	Playing different video, toggling HD on/off
Play game	Change sound, using different controls
Run flash applet	Reload page, leave page

7.2 Evaluation Criteria

To measure the effectiveness of SiPTA, we need to compare the results to different existing techniques. The prevalent metric for comparison in anomaly detection is receiver operating characteristics curve (ROC) analysis [14].

ROC Analysis. ROC analysis is a common technique in research to compare different classifiers based on their performance [27, 40]. ROC analysis explains the trade-off between the true positive rate (TPR), plotted on the y-axis, and the false positive rate (FPR), plotted on the x-axis. To compare different classifiers, the common approach [14] is to consider the classifier with the higher area under the ROC curve as the better classifier.

Figure 3 shows an ROC curve for one of the experiment runs. The algorithm calculates the values for a point *(FPR, TPR)* using Equations 12 and 13 where a classifier threshold interprets the input probabilities into a binary output of 0 (negative) or 1 (positive).

$$\mathrm{TPR} = \frac{\text{Positives correctly classified}}{\text{Total Positives}} \tag{12}$$

$$\mathrm{FPR} = \frac{\text{Negatives incorrectly classified}}{\text{Total Negatives}} \tag{13}$$

To obtain all points for the ROC curve when using Markov Model, the algorithm varies the threshold over a range of all input probabilities to obtain the corresponding points $(\mathtt{FPR}, \mathtt{TPR})$ plotted in the figure. In SiPTA, this corresponds to varying the value of θ_{M}.

In addition to the area under the ROC curve, some important characteristics of the curve help with the analysis of the classifier performance. For example, point $(0, 1)$ indicates perfect classification while the region under the dotted line $\mathtt{TPR} = \mathtt{FPR}$ indicates that the classification is worse than making a random guess.

7.3 Comparison to Alternative Approaches

To evaluate SiPTA, we compare it to other methods that implement alternative concepts. For example, first-order Markov Model [10], relies on a stochastic approach, which assumes that anomalies change the probabilistic characteristics of event transitions in a sequence of system events. Another alternative is Neural Networks [6] which trains a neural network using a set of normal and anomalous traces. The two techniques were chosen due to their popularity in the domain [9]. As discussed later in this section, the disadvantage of using Neural Networks, is the inability to use the ROC analysis to compare its performance to SiPTA, and Markov models.

7.3.1 Markov Model Technique. Markov Model is a discrete-time stochastic process used to study the probability of the change of a random variable value. First-order Markov Models are commonly used to study the probabilistic characteristics of a single transition between two events within the trace sequence [18, 38–40].

To compare to the performance of SiPTA, we implemented the anomaly detection engine shown in Figure 1 using first-order Markov Model. For an input trace sequence, an *event* represents a Markov Model state so that the Markov Model will describe the probability of occurrence of a transition between an *event* and its first successor. Following the work-flow in Figure 1, the preprocessor splits the trace into sub-traces based on the *process name* and extracts only the *event name* and *event class* attributes. For each sub-trace, the Markov Model calculates a transition probability matrix [38], which indicates the probability of transition between any trace entries. The averaged transition probability matrices calculated for the training set describe the normal behavior of the system.

In the testing phase, the classifier compares the transition probability matrix of the test trace and the normal behavior matrix to decide if the test trace is normal or anomalous. We assign an anomaly flag for each transition in the test trace that occurs with a probability value that lies outside a defined region around the mean value of probabilities that describe the normal behavior of the system. For each experiment, we performed several experiment runs using different sizes for that region to select the region size that yields the best ROC curve. For the final binary classification, the percentage of anomalous transitions within the trace indicates, whether the trace is anomalous or not. Varying a threshold over the range of the percentages, ranging from 0% to 100%, yields points of ROC curve as described earlier.

For the sake of simplicity of the implementation, transition probability matrices consider only the transitions in a randomly selected normal trace during the training phase instead of considering all possible transitions combinations. This consideration reduces the calculations by excluding the transitions that rarely occur and have no effect on the final classification result.

7.3.2 Neural Networks Technique. Artificial Neural Networks (NN) are massively connected networks of computational nodes or *neurons*. The nodes are usually organized in layers (input, output, and hidden) with weighted connections between them [20]. As the network learns, it updates the weights on the connections to improve classification.

For the purpose of comparison, the Kohonen self-organizing network (KSON) was used. The network uses unsupervised learning algorithms to cluster inputs into groups with similar characteristics. The learning is called unsupervised because the output characteristics of the network are determined internally and locally by the network itself, without any data on desired outputs. The nodes distribute themselves across the input space to recognize groups of similar input vectors, while the output nodes compete among themselves to be fired one at a time in response to a particular input vector [20]. Thus, due to this competitive learning, similar input vectors activate physically close output nodes. We want to take advantage of this characteristic of KSON to classify the traces.

The input vectors for the network are an encoded representation of the events in a trace. To generate the encoding we extract event names from the trace and then count the number of occurrences of each event. The count for each event is then scaled by dividing it by the total number of logged events in the trace. The input vector is thus a collection of event to count mappings for the trace.

The training sets for the network need to contain both clean and anomalous traces. When the network is trained with only clean traces, it is not able to classify anomalous traces as part of a different cluster during testing. Thus, unlike other approaches, Neural Networks imposes constraints on the training set. During the testing phase, the network determines the cluster that the trace belongs to and thus classifies the trace. The classification is typically discrete, with the output being either 0 (clean) or 1 (anomalous), however the value can be within that interval in case of more uncertainty.

The difficulty with using Neural Networks technique for comparison with the other approaches is the lack of a classifier threshold. The classification takes place within the internal structure of the network using specialized learning algorithms. We can thus alter the structure of the network, but cannot alter any thresholds that influence the cluster that the network will choose. As a consequence, we can only report the detection rate for this technique but cannot perform an ROC analysis.

8 RESULTS

The results of our experiments are summarized in Table 2, that shows the detection rates (TPR) and false-alarm rates (FPR) for each of the four approaches, namely SiPTA MD, SiPTA, Markov Model and Neural Networks. SiPTA MD accounts for the new implementation in this paper, for our previous work we refer the reader to [41]. Contrary to Neural Networks, SiPTA and Markov Model implement a binary classifier, this allows the use of ROC curves to compare their performance, but a similar comparison cannot be done with Neural Networks.

As mentioned in Section 7.2, the points $(\mathrm{FPR}, \mathrm{TPR})$ represent the ROC curve for each approach. Figure 3.a, 3.b, and 3.c show the ROC curves for the hexacopter, QNX-Car run *top* command, and variable ping speed scenarios, respectively. The remaining scenarios yielded ROC curves similar to Figure 3 which show near perfect classification for both SiPTA and Markov Model, with SiPTA MD improving classification performance when compared to its former implementation.

Receiver operational characteristic curves clearly demonstrate the trade-off between the detection rate and the false-alarm rate for a binary classifier. Those ROC curve points closer to $(0, 100)$ indicate better results with 100% detection rate and 0% false alarm rate. Although such points are most desirable, for our comparisons in Table 2, we favor the detection rate while tolerating false-alarm rate. For example, there are two points at equal distance to $(0, 100)$ for the Run *top* command, and SiPTA method. We report the point $(20, 100)$ as a better threshold than $(0, 80)$ because for most cases the penalty of false negatives outweighs false positives.

9 DISCUSSION

SiPTA outperforms all other approaches. In every studied case, SiPTA MD yields better results than the contrast approaches (including SiPTA). For instance, as Table 2 indicates, SiPTA yields perfect classification results for the *variable speed ping* scenario while Markov Model has 80% detection rate with 20% false alarm, and Neural Network yields a 75% detection rate. An interesting question arises from these results: Why SiPTA performs so well when other established methods fail?

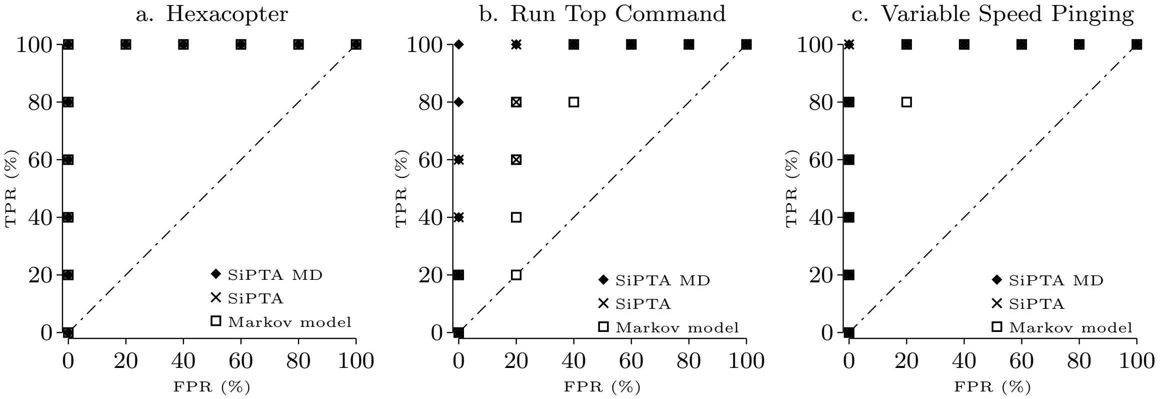


Fig. 3. Receiver operational characteristic curves for some scenarios. Each plot contain 11 points per method, each point corresponds to a different threshold configuration. Points close to coordinate $(0, 100)$ are the most desirable.

Table 2. Results summary

	SIPTA MD		SIPTA		Markov Model		Neural Network	
Scenario	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Hexacopter (I & II)	100	0	100	0	100	0	91	0
QNX-Car Idle	100	0	100	0	100	0	91	0
QNX-Car play MP3	100	0	100	0	100	0	45	0
QNX-Car Play Video	100	0	100	0	100	0	83	0
QNX-Car Run *top* command	100	0	100	20	80	20	26	0
Variable speed ping	100	0	100	0	100	20	75	0

The reason why both implementations of SiPTA work in this type of scenario is that a change in the ping rate also changes the inter-arrival time for each ping command. The overhead to the system, introduced by the demand of processing the ping commands also affects the inter-arrival time for other events within the system. The change induced in the IATS for many channels within the system will appear on the PSD as a change in the frequency location for the dominant peaks. The plots in Figure **??** show two different PSDs, the normal is on the left and the anomalous case on the right. The anomalous trace does not contain the same MPFm features when compared to the normal case.

The Markov Model method fails here because changing the ping rate does not necessarily change the sequence of events within a channel. Therefore, for some run-time conditions the transition probability matrix remain invariant to the new conditions. A transition probability matrix that remains invariant in presence of anomalous behaviour is the main reason this method may classify 20% of anomalous traces as normal. Notice that changing the detection threshold does not improve detection rate immediately, as it will start considering normal traces as anomalous. Although the technique seems to work reasonably well and is currently the dominantly used one [9], it failed to handle certain scenarios. We conjecture that Markov Model will fail whenever the anomaly has an insignificant effect on the transition probabilities. This is due to events changing their inter-arrival time without affecting their transitions probabilities, which represents a whole class of anomalies that SiPTA can detect, however, Markov Model cannot.

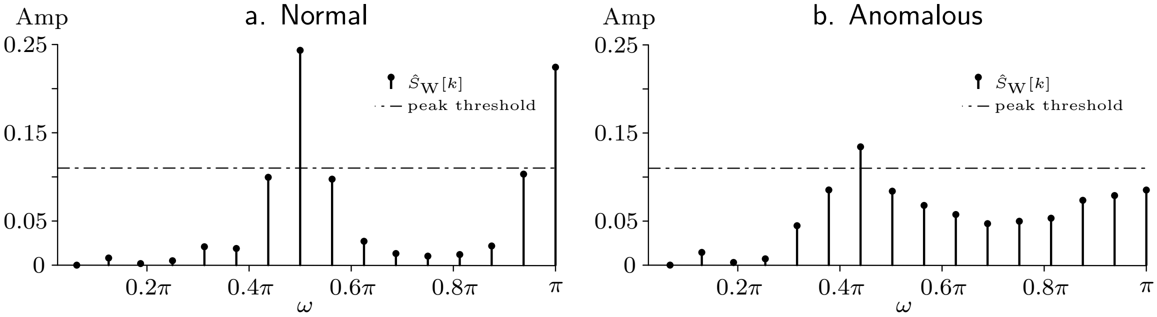


Fig. 4. In (a) the normal PSD for channel with *pid* 8204 in the variable speed ping scenario, an anomalous example is shown in plot (b).

Neural Networks are ill-suited for trace-based anomaly detection. Neural Networks is an established technique that is used for detecting anomalies, but it is less suitable for this given problem. The results for Neural Networks indicate that the technique classifies anomalous traces only for some of the scenarios. Neural Networks are only able to classify anomalous traces if trained with at least one anomaly similar to the one occurring. Table 2 shows that Neural Networks have a 0% false alarm rate throughout the experiments, because when the network is not trained with anomalous traces, then it will tend by default to classify traces as normal. Unlike SiPTA and Markov Model, Neural Networks require a different training set comprising both normal and anomalous traces.

The concepts are widely applicable. It can be observed that SiPTA has an inherently modular structure. Each component, namely trace-to-signal modeling, signal processing algorithm, metrics, and classification technique can be modified to suit application specific scenarios as long as the modules complement each other. For example, one can use some alternative method to model trace-to-signal mappings other than assigning channels to each parameter. Similarly, one can plug-in another signal processing algorithm that exploits some other domain knowledge.

Threats to validity. Our framework is based on the assumption that anomalies cause changes in the inter-arrival times of events in the trace otherwise, such anomalies will pass undetected. Therefore, this assumption needs validity verification, so SiPTA can be used whenever the assumption is valid and dropped otherwise.

Similar to Markov Models, our method still needs supervised learning from a labeled training set. Even though SiPTA was able to outperform all other studied approaches to anomaly detection for the comprehensive set of scenarios we created, more evidence is necessary. We need an extension of the empirical study to more complex scenarios and include more traces in the analysis as to gain confidence that the method will consistently outperform the contrasting approaches.

10 CONCLUSION

Identifying an incorrect behavior is crucial for safety-critical embedded systems. In this work we demonstrated that, with an appropriate application of signal processing algorithms on execution traces, one can identify an incorrect system behavior. We demonstrated the feasibility of such an approach by implementing these algorithms into SiPTA. In our experiments, we performed a holistic evaluation of SiPTA by running it on execution traces from varied execution scenarios. To demonstrate the effectiveness of SiPTA, we compared it with state of art techniques of Markov Model and Neural Networks. The results indicate that SiPTA outperforms all the studied contemporary approaches.

REFERENCES

[1] Audit data from MIT Lincolin lab. URL http://www.ll.mit.edu/mission/.

[2] QNX Neutrino RTOS. URL http://www.qnx.com/products/neutrino-rtos/neutrino-rtos.html.

[3] System call dataset from University of New Mexico. URL http://www.cs.unm.edu/~immsec/data-sets.htm.

[4] Unmanned Aerial Vehicle (UAV). URL https://uwaterloo.ca/embedded-software-group/projects/unmanned-aerial-vehicle-uav-exemplar.

[5] QNX CAR Platform for Infotainment. URL http://www.qnx.com/products/qnxcar/.

[6] A. K. Ghosh and A. Schwartzbard. A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proc. 8th USENIX Security Symposium*, pages 23–36. USENIX, 1999.

[7] M. Bartlett. The statistical approach to the analysis of time-series. *Transactions of the IRE Professional Group on Information Theory*, 1(1):81–101, Feb 1953. ISSN 2168-2690. doi: 10.1109/TIT.1953.1188570.

[8] M. S. Bartlett. Periodogram analysis and continuous spectra. *Biometrika*, 37(1/2):1–16, 1950. ISSN 00063444. URL http://www.jstor.org/stable/2332141.

[9] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[10] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

[11] X. Cheng, K. Xie, and D. Wang. Network Traffic Anomaly Detection Based on Self-Similarity Using HHT and Wavelet Transform. In *Fifth International Conference on Information Assurance and Security, IAS*, volume 1, pages 710–713. IEEE, 2009.

[12] G. Creech and J. Hu. A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns. 2013.

[13] J.L. Diaz, Kanghee Kim, José María López, and L. Lo Bello. Stochastic analysis of priority-driven periodic real-time systems. In *Handbook of Real-Time and Embedded Systems*, 2007.

[14] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[15] J. Gao, G. Hu, X. Yao, and R. K. C. Chang. Anomaly Detection of Network TrafïňĄc Based on Wavelet Packet. In *Asia-Pacific Conference on Communications. APCC*, pages 1–5. IEEE, 2006.

[16] Gabriella Gigante and Domenico Pascarella. *Formal Methods in Avionic Software Certification: The DO-178C Perspective*, pages 205–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-34032-1. doi: 10.1007/978-3-642-34032-1_21. URL https://doi.org/10.1007/978-3-642-34032-1_21.

[17] T. Herpel, B. Kloiber, R. German, and S. Fey. Assessing the can communication startup behavior of automotive ecus by prototype measurements. In *2009 IEEE Instrumentation and Measurement Technology Conference*, pages 928–932, May 2009. doi: 10.1109/IMTC.2009.5168584.

[18] S. Jha, K. MC. Tan, and R. A. Maxion. Markov Chains, Classifiers, and Intrusion Detection. In *csfw*, volume 1. Citeseer, 2001.

[19] Peter Kafka. The automotive standard iso 26262, the innovative driver for enhanced safety assessment & technology for motor cars. *Procedia Engineering*, 45:2 – 10, 2012. ISSN 1877-7058. doi: http://dx.doi.org/10.1016/j.proeng.2012.08.112. URL http://www.sciencedirect.com/science/article/pii/S1877705812031244. 2012 International Symposium on Safety Science and Technology.

[20] F.O. Karray and C. De Silva. *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*, volume 1. Pearson Education Limited, 2004.

[21] S. Kay. *Intuitive Probability and Random Processes using MATLAB®*. Intuitive Probability and Random Processes Using MATLAB. Springer US, 2006. ISBN 9780387241579. URL https://books.google.ca/books?id=aSjX2tTpmFcC.

[22] Philip A Laplante. *Real-Time Systems Design and Analysis. Third Edition.* IEEE Press and Wiley-Interscience, 2004. ISBN 0-471-22855-9.

[23] C L Liu and J W Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[24] W. Lu and A. A. Ghorbani. Network Anomaly Detection Based on Wavelet Analysis. *EURASIP Journal on Advances in Signal Processing*, 2009, 2009.

[25] Sorin Manolache, Petru Eles, and Zebo Peng. *Real-Time Applications with Stochastic Task Execution Times: Analysis and Optimisation.* Springer Science & Business Media, 2007.

[26] D.G. Manolakis and V.K. Ingle. *Applied Digital Signal Processing: Theory and Practice.* Cambridge University Press, 2011. ISBN 9780521110020. URL https://books.google.ca/books?id=XJnWngEACAAJ.

[27] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture. A Host-based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules. 2013.

[28] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing.* Pearson Education, 2011. ISBN 9780133002287. URL https://books.google.ca/books?id=EaMuAAAAQBAJ.

[29] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. And Intelligent Manufacturing Systems. Prentice Hall, 1996. ISBN 9780133737622. URL https://books.google.ca/books?id=sAcfAQAAIAAJ.

[30] S. Rawat and C. S. Sastry. Network Intrusion Detection Using Wavelet Analysis. In *Intelligent Information Technology*, pages 224–232. Springer, 2005.

[31] M. Salagean and I. Firoiu. Anomaly Detection of Network Traffic Based on Analytical Discrete Wavelet Transform. In *8th International Conference on Communications (COMM)*, pages 49–52. IEEE, 2010.

[32] Arthur Schuster. On the investigation of hidden periodicities with application to a supposed 26 day period of meteorological phenomena. *Terrestrial Magnetism*, 3(1):13–41, 1898. ISSN 0272-7528. doi: 10.1029/TM003i001p00013. URL http://dx.doi.org/10.1029/TM003i001p00013.

[33] Lui Sha, Tarek F. Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore P. Baker, Alan Burns, Giorgio C. Butazzo, Marco Caccamo, John P. Lehoczky, and Aloysius K Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28(2):101–155, 2004. ISSN 0922-6443.

[34] Eg3 Team. Selecting an Embedded RTOS. Technical report, EG3, 2008.

[35] Michail Vlachos, Philip Yu, and Vittorio Castelli. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 449–460. SIAM, 2005.

[36] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, Jun 1967. ISSN 0018-9278. doi: 10.1109/TAU.1967.1161901.

[37] P. Welch. A fixed-point fast fourier transform error analysis. *IEEE Transactions on Audio and Electroacoustics*, 17(2): 151–157, Jun 1969. ISSN 0018-9278. doi: 10.1109/TAU.1969.1162035.

[38] N. Ye and X. Li. A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, pages 171–174. Oakland: IEEE, 2000.

[39] N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu. Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31(4):266–274, 2001.

[40] N. Ye, Y. Zhang, and C. M. Borror. Robustness of the Markov-Chain Model for Cyber-Attack Detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.

[41] Mohammad Mehdi Zeinali Zadeh, Mahmoud Salem, Neeraj Kumar, Greta Cutulenco, and Sebastian Fischmeister. SiPTA: Signal processing for trace-based anomaly detection. In *Proceedings of the 14th International Conference on Embedded Software*, page 6. ACM, 2014.

[42] M. Zhou and S. D. Lang. Mining Frequency Content of Network Traffic for Intrusion Detection. In *Proceedings of the IASTED International Conference on Communication, Network, and Information Security*, 2003.