

FAULT DIAGNOSIS FOR IP-BASED NETWORK WITH REAL-TIME CONDITIONS



ANGELA MARÍA VARGAS ARCILA

Doctoral Thesis
Ph.D. in Telematics Engineering
Ph.D. in Computer Science and Technology

Research advisors:

Ph.D. Araceli Sanchis de Miguel (UC3M)
Ph.D. Alvaro Rendón Gallón (Unicauca)

Tutor:

Ph.D. Juan Carlos Corrales (Unicauca)

Universidad del Cauca
Universidad Carlos III de Madrid

September 2021

ANGELA MARÍA VARGAS ARCILA

**FAULT DIAGNOSIS FOR IP-BASED NETWORK WITH
REAL-TIME CONDITIONS**

Thesis carried out in co-supervision between Universidad del Cauca (Unicauca)
and Universidad Carlos III de Madrid (UC3M)
for obtaining the title of

**PhD in Telematics Engineering
PhD in Computer Science and Technology**

Research advisors:

Ph.D. Araceli Sanchis de Miguel (UC3M)

Ph.D. Alvaro Rendón Gallón (Unicauca)

Tutor:

Ph.D. Juan Carlos Corrales (Unicauca)

Popayán (Colombia) / Madrid (Spain)

2021

Esta tesis se distribuye bajo licencia “Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**”.



*A mi mamá y mi papá,
por su infinita lealtad y amor incondicional.*

Agradecimientos

Hacer un doctorado es emprender un viaje sin rumbo fijo, sin haber trazado una ruta. En mi caso, tuve un objetivo, pero no tenía claro cómo llegar a él o si era posible llegar a él, así que empecé a caminar, a explorar y a convertirme en una persona errante. Lo peligroso de este viaje, es que puedes vagar indefinidamente y extraviarte en el camino, no solo en la ruta de la tesis, sino en la vida misma, en el quién soy y para qué deseo esto. Mi viaje, además, no tuvo autopistas ni vuelos directos, recorrí carreteras destapadas, subí montañas y cerros peñascosos, y me encontré cansada, con abismos desde donde quise saltar; sin embargo, continué con la esperanza de llegar a un lugar donde me sintiera a salvo y pudiera terminar mi recorrido.

Emprendí sola mi viaje, pero llevé cargada mi mochila con todos los elementos necesarios para sobrevivir. Lo más importante, mi seguro contra todo riesgo: mi papá y mi mamá, a quienes agradezco infinitamente por su apoyo incondicional y por lanzarme salvavidas cuando más lo necesitaba... ya sabemos que un viaje doctoral lo inicias con los tenis puestos y terminas a pie limpio. A mi energizante: Oscar, por escuchar todas las decepciones que sentí frente a la academia, por ayudar a tranquilizarme y volver a ver el lado bueno de mi travesía, por no dejarme desfallecer. A mis brújulas: Araceli Sánchez y Alvaro Rendón, porque su dirección fue indispensable en este proceso. No voy a olvidar el cafecito con leche que la profe me brindó cuando quería tirar la toalla, ni los impresionantes racionamientos que el ingeniero hizo alrededor de mis ideas.

A quienes me crucé en el viaje y con quienes recorrí trozos de mi camino: mis compañeros en la UC3M y Unicauca. Gracias a Yosef, Julián y Elenice por su compañía, fuimos el toque multicultural y desfasado (en horario) del laboratorio CAOS. A Maicol, Arturo, y Claudia que se sumaron a nosotros y

convirtieron la hora del almuerzo en un espacio de distensión y risas. Antes de compartir con ustedes me creía una persona cuerda dentro de los estándares colombianos, sin embargo, ver el mundo con su compañía me hizo comprender que esa cordura sólo es válida para un entorno; cada uno de ustedes tiene una particularidad, algo de locura desde mi punto de vista, pero si cada uno de nosotros piensa eso de los demás, eso me hace loca desde la orilla de otros, desde lugares y mundos lejanos al mío. En Unicauca, agradezco a la cálida oficina 116: Pipe, Angelita, Carlos, Fulvio, Julián y Maritza, gracias por su conocimiento, por compartir alegrías y, sobretodo, preocupaciones y tristezas; fue un gran apoyo escuchar sus opiniones, valoro su criterio y saber que no estaba tan sola en mi recorrido.

A quienes coincidí o conviví durante estos años, a quienes me abrieron las puertas de su casa para invitarme a un almuerzo, a quienes fueron conmigo a lugares desconocidos...

Muchas gracias, creo que ahora soy un buen resultado de muchas coincidencias.

Published and submitted content

- Vargas Arcila, A. M., Corrales Muñoz, J. C., Rendon Gallon, A., & Sanchis, A. (2021). Selection of Online Network Traffic Discriminators for on-the-Fly Traffic Classification. *Revista Ingenierías*, 20(38), 65-85. <https://doi.org/10.22395/rium.v20n38a4>
 - This work is not included in the thesis.
- Vargas-Arcila, A.M., Corrales, J.C., Sanchis, A., & Rendón Gallón, A. Peripheral Diagnosis for Propagated Network Faults. *Journal of Network and Systems Management* volume 29, 14 (2021). <https://doi.org/10.1007/s10922-020-09579-0>
 - This work is totally included in the thesis (in Introduction, some sections of Chapter I, II, III, and IV).
- Vargas-Arcila, A. M., Corrales, J. C., Sanchis, A., & Rendón, A. (2021). *Dataset of symptom-fault causal relationships for an IP-based network*. Mendeley Data, V1. <https://doi.org/10.17632/tc6ysmh5j8.1>
 - This work is partially included in the thesis (in Chapter IV and indicated by explicit reference).
- Vargas-Arcila, A. M., Corrales, J. C., Sanchis, A., & Rendón, A. *SOFI dataset: Symptom-Fault relationship for IP-Network*. Submitted in: *Computer Networks*.
 - This work is totally included in the thesis (in Chapter IV).

Structured Abstract

BACKGROUND:

Fault diagnosis techniques have been based on many paradigms, which derive from diverse areas and have different purposes: obtaining a representation model of the network for fault localization, selecting optimal probe sets for monitoring network devices, reducing fault detection time, and detecting faulty components in the network. Although there are several solutions for diagnosing network faults, there are still challenges to be faced: a fault diagnosis solution needs to always be available and able enough to process data timely, because stale results inhibit the quality and speed of informed decision-making. Also, there is no non-invasive technique to continuously diagnose the network symptoms without leaving the system vulnerable to any failures, nor a resilient technique to the network's dynamic changes, which can cause new failures with different symptoms.

AIMS:

This thesis aims to propose a model for the continuous and timely diagnosis of IP-based networks faults, independent of the network structure, and based on data analytics techniques.

METHOD(S):

This research's point of departure was the hypothesis of a fault propagation phenomenon that allows the observation of failure symptoms at a higher network level than the fault origin. Thus, for the model's construction, monitoring data was collected from an extensive campus network in which impact link failures were induced at different instants of time and with different duration. These data correspond to widely used parameters in the actual management of a network. The collected data allowed us to understand the faults' behavior and how they are manifested at a peripheral level.

Based on this understanding and a data analytics process, the first three modules of our model, named PALADIN, were proposed (Identify, Collection and Structuring), which define the data collection peripherally and the necessary data pre-processing to obtain the description of the network's state at a given moment. These modules give the model the ability to structure the data considering the delays of the multiple responses that the network delivers to a single monitoring probe and the multiple network interfaces that a peripheral device may have.

Thus, a structured data stream is obtained, and it is ready to be analyzed. For this analysis, it was necessary to implement an incremental learning framework that respects networks' dynamic nature. It comprises three elements, an incremental learning algorithm, a data rebalancing strategy, and a concept drift detector. This framework is the fourth module of the PALADIN model named Diagnosis.

In order to evaluate the PALADIN model, the Diagnosis module was implemented with 25 different incremental algorithms, ADWIN as concept-drift detector and SMOTE (adapted to streaming

scenario) as the rebalancing strategy. On the other hand, a dataset was built through the first modules of the PALADIN model (SOFI dataset), which means that these data are the incoming data stream of the Diagnosis module used to evaluate its performance.

The PALADIN Diagnosis module performs an online classification of network failures, so it is a learning model that must be evaluated in a stream context. Prequential evaluation is the most used method to perform this task, so we adopt this process to evaluate the model's performance over time through several stream evaluation metrics.

RESULTS:

This research first evidences the phenomenon of impact fault propagation, making it possible to detect fault symptoms at a monitored network's peripheral level. It translates into non-invasive monitoring of the network. Second, the PALADIN model is the major contribution in the fault detection context because it covers two aspects. An online learning model to continuously process the network symptoms and detect internal failures. Moreover, the concept-drift detection and rebalance data stream components which make resilience to dynamic network changes possible. Third, it is well known that the amount of available real-world datasets for imbalanced stream classification context is still too small. That number is further reduced for the networking context. The SOFI dataset obtained with the first modules of the PALADIN model contributes to that number and encourages works related to unbalanced data streams and those related to network fault diagnosis.

CONCLUSIONS:

The proposed model contains the necessary elements for the continuous and timely diagnosis of IP-based network faults; it introduces the idea of periodical monitorization of peripheral network elements and uses data analytics techniques to process it. Based on the analysis, processing, and classification of peripherally collected data, it can be concluded that PALADIN achieves the objective. The results indicate that the peripheral monitorization allows diagnosing faults in the internal network; besides, the diagnosis process needs an incremental learning process, concept-drift detection elements, and rebalancing strategy.

The results of the experiments showed that PALADIN makes it possible to learn from the network manifestations and diagnose internal network failures. The latter was verified with 25 different incremental algorithms, ADWIN as concept-drift detector and SMOTE (adapted to streaming scenario) as the rebalancing strategy.

This research clearly illustrates that it is unnecessary to monitor all the internal network elements to detect a network's failures; instead, it is enough to choose the peripheral elements to be monitored. Furthermore, with proper processing of the collected status and traffic descriptors, it is possible to learn from the arriving data using incremental learning in cooperation with data rebalancing and concept drift approaches. This proposal continuously diagnoses the network symptoms without leaving the system vulnerable to failures while being resilient to the network's dynamic changes.

PRIMARY KEYS: network management, network monitoring, fault diagnosis, incremental learning, data stream.

Content

INTRODUCTION	1
1 OBJECTIVES.....	2
2 RESEARCH CONTRIBUTIONS.....	2
3 DOCUMENT STRUCTURE	3
CHAPTER I: BACKGROUND	4
1 IP NETWORK	4
2 FAULTS, FAILURES, AND SYMPTOMS.....	6
2.1 TYPES OF NETWORK FAILURES	6
2.2 FAULT DIAGNOSIS	7
3 DATA STREAM	8
4 MACHINE LEARNING	9
5 INCREMENTAL LEARNING	13
CHAPTER II: STATE OF THE ART	15
1 NETWORK FAULT DIAGNOSIS THROUGH YEARS	15
1.1 METHODOLOGY.....	15
1.2 ANALYSIS OF RESULTS	19
1.3 FIRST PERIOD (2001-2005)	19
1.4 SECOND PERIOD (2006-2010)	20
1.5 THIRD PERIOD (2011-2015).....	21
1.6 FOURTH PERIOD (2016-2020)	22
2 NETWORK FAULT DIAGNOSIS TECHNIQUES.....	24
3 CURRENT FAULT DIAGNOSIS GAPS.....	25
4 PROBLEM STATEMENT.....	26
CHAPTER III: PALADIN - A PERIPHERAL FAULT DIAGNOSIS MODEL FOR IP-BASED NETWORKS	27
1 IDENTIFY	28
2 COLLECTION.....	28

3	STRUCTURING.....	30
4	DIAGNOSIS	31
4.1	HOW TO TACKLE IMBALANCEMENT AND CONCEPT-DRIFT	32
4.2	DIAGNOSIS COMPONENTS	34
CHAPTER IV: DATASET CONSTRUCTION		36
1	DATA UNDERSTANDING.....	37
1.1	FAULT DATA COLLECTION AND DESCRIPTION	37
1.1.1	<i>Network scenario.....</i>	<i>37</i>
1.1.2	<i>Data collection experiment.....</i>	<i>43</i>
1.1.3	<i>Data extraction and description</i>	<i>45</i>
1.2	RAW DATA EXPLORING	46
1.2.1	<i>Parameter delays.....</i>	<i>46</i>
1.2.2	<i>Late triggering of internal events</i>	<i>49</i>
1.3	DATA QUALITY VERIFICATION	50
2	DATA PREPARATION: STRUCTURING.....	50
2.1.1	<i>Time-based grouping.....</i>	<i>51</i>
2.1.2	<i>Interface-based grouping</i>	<i>54</i>
2.1.3	<i>Event correlation or labeling</i>	<i>56</i>
3	SOFI DATASET SUMMARY	57
CHAPTER V: EXPERIMENTS		59
1	STRUCTURED DATA EXPLORING	59
2	A BRIEF EXPERIMENTAL ANALYSIS FOR TRADITIONAL MACHINE LEARNING MODELS WITH THE SOFI DATASET.....	64
2.1	EXPERIMENTAL SETUP	64
2.2	PERFORMANCE EVALUATION METRICS	65
2.3	EXPERIMENTAL RESULTS AND DISCUSSION	68
3	PALADIN MODEL EVALUATION	71
3.1	EXPERIMENTAL SETUP	72
3.2	EVALUATION METHOD AND IMBALANCED STREAM EVALUATION METRICS.....	75
3.3	EXPERIMENTAL RESULTS AND DISCUSSION	77
CHAPTER VI: CONCLUSION AND FUTURE WORK		85
ANNEXES.....		88
ANNEX A: NETWORK EMULATION ON GNS3.....		88
A.1	ACCESS LAYER CONFIGURATION	88
A.1.1	<i>Access Switches Configuration</i>	<i>88</i>
A.1.2	<i>Clients Configuration</i>	<i>92</i>
A.2	DISTRIBUTION LAYER CONFIGURATION	93
A.3	CORE LAYER CONFIGURATION	100
A.3.1	<i>Core Switches Configuration.....</i>	<i>100</i>
A.3.2	<i>Firewall Configuration</i>	<i>101</i>
A.4	SERVER FARM CONFIGURATION.....	107
A.4.1	<i>Switch Configuration</i>	<i>107</i>
A.4.2	<i>Server A Configuration.....</i>	<i>108</i>
A.4.3	<i>Server B Configuration.....</i>	<i>115</i>

A.5	ENTERPRISE EDGE AND SERVICE PROVIDER CONFIGURATION	123
A.5.1	<i>Router EDGE Configuration</i>	123
A.5.2	<i>ISP Configuration</i>	126
A.6	CREDENTIALS SUMMARY	127
ANNEX B:	DESCRIPTION OF RAW DATA	129
ANNEX C:	EXTRACTION APPLICATION FOR ZABBIX	134
C.1	SOFI RAW DATASET EXTRACTION PROCESS	135
C.2	INSTANCE GROUPING PROCESS	135
C.3	COLUMN OPERATION PROCESS	136
C.4	LABELING PROCESS	137
C.5	OTHER FUNCTIONS	138
C.6	ZABBIX DATABASE	138
ANNEX D:	EXPERIMENTS CHARTS	140
D.1	PREQUENTIAL AUC CHARTS	141
D.2	GEOMETRIC MEAN (G-MEAN) CHARTS	144
D.3	COHEN'S KAPPA COEFFICIENT CHARTS	148
D.4	MATTHEWS CORRELATION COEFFICIENT (MCC) CHARTS	151
D.5	RECALL OR SENSITIVITY CHARTS	155
REFERENCES	159

Figures

CHAPTER I

Figure 1.1 Enterprise Composite Network Model.....	5
Figure 1.2 The relation between failures and their root causes and effects.....	6
Figure 1.3 Link Failures types.	7
Figure 1.4 Fault management process.	7
Figure 1.5 Concept-drift types according to the influence on learned decisions or classification. Based on the image provided by (Fernández et al., 2018a).	9
Figure 1.6 Concept-drift types according to speed of changes taking place within the stream. Based on the image provided by (Fernández et al., 2018a).	9
Figure 1.7 Multi-Layer Perceptron Representation.....	11
Figure 1.8 AdaBoost algorithm.....	12
Figure 1.9 Example of a decision tree.	13

CHAPTER II

Figure 2.1 Workflow of bibliometric analysis with SciMAT.....	15
Figure 2.2 Evolution map appearance.....	17
Figure 2.3 Strategic diagram appearance.....	18
Figure 2.4 Cluster' network appearance.	18
Figure 2.5 Strategic map for period 2001-2005.	19
Figure 2.6 Strategic map for period 2006-2010.	20
Figure 2.7 Strategic map for period 2011-2015.	22
Figure 2.8 Strategic map for period 20016-2020.	23
Figure 2.9 Fault diagnosis techniques.	24

CHAPTER III

Figure 3.1 PALADIN - Fault diagnosis model for IP-based networks.	27
Figure 3.2 Elements of monitored network.	28
Figure 3.3 Classification of network data collection approaches, according to (Zhou et al., 2018).	29
Figure 3.4 Traditional monitorization (left) vs peripheral monitorization (right).	29
Figure 3.5 Structuring conditions.	30
Figure 3.6 Structuring steps.	30
Figure 3.7 Diagnosis conditions.....	31
Figure 3.8 The ecosystem of challenges to be faced in diagnosis.	32
Figure 3.9 Strategies to deal with concept-drift. (Bifet et al., 2018).....	33
Figure 3.10 Diagnosis components (S. Wang et al., 2013).	34

CHAPTER IV

Figure 4.1. Phases of the CRISP-DM reference model.	36
Figure 4.2. Data understanding activities.	37
Figure 4.3 Topology of the Emulated Enterprise Network.	38
Figure 4.4 Emulation Requirements.	42
Figure 4.5 Network views.	45
Figure 4.6 Elements involved in network event logs collection and network traffic data collection.	45
Figure 4.7 Structure of the SNMP extracted data.	46
Figure 4.8 Expected periodic arrival of parameters.	47
Figure 4.9 Observerd arrival of parameters over time.	47
Figure 4.10 SOFI raw dataset structure.	48
Figure 4.11 Analysis of the obtained matrix of parameter values.	48
Figure 4.12 Behavior of induced fault.	50
Figure 4.13 Steps for structuring SOFI dataset.	51
Figure 4.14 Functions for structuring SOFI dataset.	51
Figure 4.15 Parameter distribution in one polling (example 1).	53
Figure 4.16 Parameter distribution in one polling (example 2).	54
Figure 4.17 Interface grouping scheme.	55
Figure 4.18 SOFI labeling approach.	57

CHAPTER V

Figure 5.1 Range parameter behavior at different time intervals.	61
Figure 5.2 ICMP_response_time parameter behavior before, during, and after the failure. The failure time interval is indicated in red.	61
Figure 5.3 P_Bits_received and P_Bits_sent parameters behavior before, during, and after the failure. The failure time interval is indicated in red.	62
Figure 5.4 Device_uptime parameter behavior before, during, and after the failure. The failure time interval is indicated in red.	62
Figure 5.5 E_inbound_packets_with_errors parameter behavior before, during, and after the failure. The failure time interval is indicated in red.	63
Figure 5.6 E_Bits_received, I_Bits_received, E_Bits_sent and I_Bits_sent parameters behavior before, during, and after the failure. The failure time interval is indicated in red.	63
Figure 5.7 Synthetic Minority Oversampling Technique (SMOTE).	65
Figure 5.8 Confusion matrix structure.	66
Figure 5.9 Traditional classification performance.	71
Figure 5.10 Flowchart of RebalanceStream meta-strategy (Bernardo, 2019).	72
Figure 5.11 How the ADWIN drif-detector works.	73
Figure 5.12 Prequential evaluation process.	76
Figure 5.13 Performance curves format.	77
Figure 5.14 Weighted difference elements.	81
Figure 5.15 The color scale for evaluation metrics.	82
Figure 5.16 Heatmap of prequential Sensitivity/Recall results.	82
Figure 5.17 Heatmap of prequential G-mean results.	82
Figure 5.18 Heatmap of prequential Kappa statistic results.	82
Figure 5.19 Heatmap of prequential MCC results.	83
Figure 5.20 Heatmap of prequential AUC results.	83

ANNEX A

Figure A.1 Emulated network in GNS3 platform.	89
Figure A.2 Configuration of Cloud element with the loopback interface.....	93
Figure A.3 Distribution switch command output: show sflow interfaces.	99
Figure A.4 Distribution switch command output: show sflow.	99
Figure A.5 rsyslog-mysql configuration (step 1).	117
Figure A.6 rsyslog-mysql configuration (step 2).	118
Figure A.7 rsyslog-mysql configuration (step 3).	118
Figure A.8 Syslog scheme in MySQL workbench.	118
Figure A.9 rsyslog user in MySQL workbench.	119
Figure A.10 LogAnalyzer installation (error at the first start).	120
Figure A.11 LogAnalyzer installation (step 1).	120
Figure A.12 LogAnalyzer installation (step 2).	121
Figure A.13 LogAnalyzer installation (step 3).	121
Figure A.14 LogAnalyzer installation (step 4).	121
Figure A.15 LogAnalyzer installation (step 5).	122
Figure A.16 LogAnalyzer installation (step 6).	122
Figure A.17 LogAnalyzer installation (step 7).	122

ANNEX C

Figure C.1 Modules of SOFIApp extraction application.....	134
Figure C.2 SOFI raw dataset extraction process.	135
Figure C.3 Instance grouping process.....	136
Figure C.4 Format of the operations list file.....	136
Figure C.5 Column operation process.	137
Figure C.6 Format of the failure list file.....	137
Figure C.7 SOFI labeling process.....	138
Figure C.8 Tables from the Zabbix database used by SOFIApp.	139

ANNEX D

Figure D.1 Performance curves format.	140
--------------------------------------------	-----

Tables

CHAPTER I

Table 1.1 Traffic beffore and during link failure.....	7
--------------------------------------------------------	---

CHAPTER III

Table 3.1 Parameters to collect.....	29
--------------------------------------	----

CHAPTER IV

Table 4.1 Network elements of enterprise campus module.....	40
Table 4.2 Network elements of enterprise edge module.	41
Table 4.3 Network elements of service provider edge module.	41
Table 4.4 Memory allocation for emulated devices.....	42
Table 4.5 Faults to induce.	44
Table 4.6 Operations for each subtype of traffic descriptors.....	56
Table 4.7 SOFI attributes list.	58

CHAPTER V

Table 5.1 Parameter's findings details.	59
Table 5.2 Classification performance with original SOFI dataset (SwitchCore-I file)	69
Table 5.3 Classification performance with original SOFI dataset (SwitchCore-II file)	69
Table 5.4 Classification performance with undersampled SOFI dataset (SwitchCore-I file)	69
Table 5.5 Classification performance with undersampled SOFI dataset (SwitchCore-II file)	69
Table 5.6 Classification performance with oversampled SOFI dataset (SwitchCore-I file).....	70
Table 5.7 Classification performance with oversampled SOFI dataset (SwitchCore-II file).....	70
Table 5.8 List of online classifiers for evaluating the PALADIN diagnosis module.	74
Table 5.9 Some results of prequential Sensitivity/Recall.	78
Table 5.10 Some results of prequential G-Mean.	78
Table 5.11 Some results of prequential Kappa statistic.	79
Table 5.12 Some results of prequential MCC.....	79
Table 5.13 Some results of prequential AUC.....	80

ANNEX A

Table A.1 Distribution networks.....	94
Table A.2 Credentials for Zabbix appliance.	115
Table A.3 Credentials summary for emulated network elements.	127

ANNEX B

Table B.1 Parameters collected from nProbe flow files.	129
-------------------------------------------------------------	-----

Table B.2 Parameters collected from Rsyslog data base.....	132
Table B.3 Parameters collected from Zabbix data base.	133

ANNEX C

Table C.1 Description of the tables from the Zabbix database used by SOFIApp.	138
------------------------------------------------------------------------------------	-----

ANNEX D

Table D.1 Charts of results of the Prequential AUC performance metric.	141
Table D.2 Charts of results of the prequential GMean performance metric.	144
Table D.3 Charts of results of the prequential Kappa statistic performance metric.	148
Table D.4 Charts of results of the prequential Matthews Correlation Coefficient performance metric.	151
Table D.5 Charts of results of the prequential Sensitivity/Recall performance metric.	155

*“Hay que poner un gran signo de interrogación sobre el valor de lo fácil;
no solamente sobre sus consecuencias, sino sobre la cosa misma,
sobre la predilección por todo aquello que no exige de nosotros ninguna superación,
ni nos pone en cuestión, ni nos obliga a desplegar nuestras posibilidades”.*

*(Estanislao Zuleta
in his discourse accepting the Honorary Doctorate Degree
awarded by Universidad del Valle, Colombia 1980)*

INTRODUCTION

Networks have dramatically increased in size and complexity over the years. The increase in complexity presents severe challenges to the operations of network management systems. One of the central issues faced by a network management system is fault management. Failures are unavoidable in large communication networks, so the timely detection and identification of failures are vital for the reliable operation of the networks (Dusia & Sethi, 2016a).

In the field of fault management, there are three basic concepts: faults, errors, and symptoms. Faults constitute a class of network events that can cause other events but are not themselves caused by other events. Error is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition. The errors are a consequence of a fault, and they may cause deviation of a delivered service which is visible to the outside world. The term failure denotes this type of error. Finally, symptoms are external manifestations of failures (Steinder & Sethi, 2004c). The fault detection process in networking aims to identify if failures have occurred by observing network symptoms (Dusia & Sethi, 2016a).

In the past, many paradigms were proposed upon which fault diagnosis techniques were based. These paradigms derive from different areas (e.g., computer science, including artificial intelligence, graph theory, neural networks, machine learning, statistical analysis) and have different purposes: obtain a representation model of the network for fault localization, selecting optimal probe sets for monitoring network devices, reduce fault detection time, and detection of faulty components in the network, among others.

So, there are a variety of fault detection techniques. The most traditional ones are passive monitoring techniques, active monitoring techniques, decentralized probabilistic management approaches, and temporal correlation techniques. However, they are invasive because of the increase in network traffic and control overhead. On the other hand, techniques for overlay and virtual networks increase the internal processes of the network because of the need to install monitoring agents on all overlay nodes. This last problem also applies to decentralized management techniques because of the need to have an embedded management process on all networking devices.

All the above techniques need in-depth knowledge of the network connectivity and operations in addition to an extensive understanding of network behavior. So, to solve it, fault diagnosis based on machine learning techniques emerge.

Learning techniques would not require a complete dependency model of a network (Dusia & Sethi, 2016a). For example, a network management system that monitors complex networks can generate a comprehensive history of *SNMP* (Simple Network Management Protocol) requests or a large number of log files. So, using statistical and machine learning techniques to process this information, it can get empirical data.

Nevertheless, existing learning techniques have some gaps. They deal with insufficient data samples, and lengthy retraining may be required whenever the system behavior changes significantly. Moreover, these techniques have to be thrown away during the period of retraining, leaving the systems vulnerable to any failures. On the other hand, a fault diagnosis solution needs to be capable enough to timely process data because stale results inhibit the quality and speed of informed decision-making.

In brief, there isn't a non-invasive technique to continuously diagnose the network symptoms without leaving the system vulnerable to any failures, nor a resilient technique to the dynamic changes of the network, which can cause new failures with different symptoms.

Consequently, this work proposes a fault detection model based on the peripheral observation of failure symptoms, which are processed continuously and timely by incremental learning algorithms. In this way, a non-invasive and resilient fault detection to the dynamic changes of the network will be guaranteed.

1 Objectives

This thesis aims to propose a model for the continuous and timely diagnosis of IP-based networks faults, independent of the network structure, and based on data analytics techniques.

Specific Objectives:

1. Create a data set that relates peripheral symptoms with network faults.
2. Propose an online fault classification model with data stream conditions.
3. Develop and evaluate the fault diagnosis model through an experiment that measures its behavior over time.

2 Research Contributions

This doctoral research makes the following contributions:

- A formal classification of link failures, which is found in Chapter I.
- A peripheral strategy for collecting failure symptoms reflected in Chapter III.
- The PALADIN model to detect failures in IP networks, which represents the overall doctoral proposal. It is described in Chapter III.
- The introduction to the concept "*peripheral network element*" for monitored networks through the proposed approach.
- The SOFI dataset (Symptom-Fault relationship for IP-Network) and its building guide, described in Chapter IV.

- A network fault diagnosis module based on symptoms and incremental learning techniques without in-depth knowledge of the network. It is described in Chapter III.
- Pieces of evidence of the existence of the propagation phenomenon for impact link-failures. Chapter V explains them.

3 Document Structure

This document is divided as follows. Chapter I describes the relevant concepts for the work. Chapter II presents a state-of-the-art study about fault diagnosis techniques used over time. Chapter III describes the proposed model. Chapter IV describes the dataset construction, which was built based on three components of the proposed model. Chapter V contains the experiments that support the research results. Finally, Chapter VI presents the conclusions and future work.

Chapter I: Background

*“Nothing in life is to be feared, it is only to be understood.
Now is the time to understand more,
so that we may fear less.”
(Marie Curie)*

This chapter presents the essential concepts of this thesis proposal development. Firstly, the IP Network, which depicts the context for which the proposal is developed. Secondly, the fault, failures, and symptoms terms that represent the problem area to face in that context. Thirdly, the machine learning paradigm, along with the descriptions of the algorithms discussed later in experiments. Finally, this chapter describes the incremental learning approach, which is our way of solving the problem, and the algorithms' list involved in the thesis evaluation.

1 IP Network

An IP network is a communication network whose elements (hosts or network nodes) use the Internet Protocol (IP) to send and receive messages between them. Its design follows a top-down approach, loosely using the TCP/IP stack. The Internet is the best known and largest IP network.

The IP network topology depends on its size. This work is focused on large networks, which have a modular design known as the Enterprise Composite Network Model. The modular network design separates the network into functional areas or major modules, each targeting a specific place and purpose in the network, and also facilitates the design of larger, more scalable networks because the model follows the rules of hierarchical network design (Cisco Academy, 2014).

As shown in Figure 1.1, the enterprise network model has three main modules: enterprise campus, enterprise edge, and service provider edge. Each of them is further divided to define specific functions for the network.

The Service Provider Edge module represents the physical connection terminations with several service providers. It is necessary for enabling communication with other networks through Internet Service Providers and WAN technologies (Conlan, 2009).

The Enterprise Edge module allows the connectivity between elements outside the campus and routes the traffic into the campus core. This connection is made directly or through an optional Edge Distribution module, which is the last line of defense against external attacks (Teare, 2012).

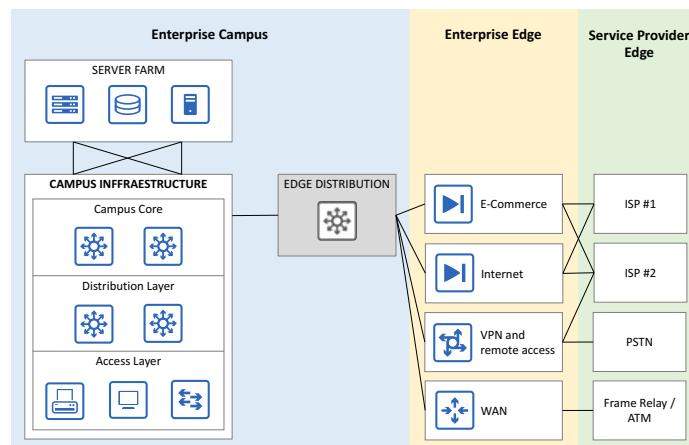


Figure 1.1 Enterprise Composite Network Model.

The Enterprise Campus module is a large site, usually a corporate headquarter or a head office, which houses all the local area networks, such as regional offices, SOHOs (Small Office, Home Office), and mobile workers (Teare, 2012) (Conlan, 2009). It is composed of the submodules: campus infrastructure and server farm.

The campus infrastructure provides network access to end users. The network of this module adheres to the classical hierarchical design model made up of three layers: the access, distribution, and core layers (Conlan, 2009). It spans from a single building to several buildings for larger enterprises connected across a core. A single building contains an access and distribution layer. When more buildings are added, a backbone or core layer is added between buildings (Teare, 2012). Following, each of these layers is described:

Access layer: It provides users or end devices (including teleworkers or remote sites for WAN environments) access to the corporate network and uplinks to the distribution layer (Cisco Academy, 2014).

Distribution layer: It receives the data from the access layer switches to transmit them to the core layer for routing to the final destination. The above means it is the boundary between the access layer and the core layer (Cisco Academy, 2014).

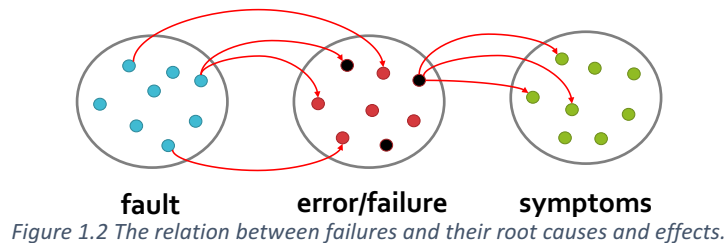
Core layer: It is also known as the network backbone and is the central point in the network. The campus core layer must be highly available and redundant because it interconnects the distribution layer with the server farm and the enterprise edge modules. It carries all the traffic from these modules, so it also must forward a large amount of data as quickly and efficiently as possible.

Finally, the server farm module follows a multilayer architecture as the campus infrastructure, and hosts in the access layer all the servers which deploy the applications that the users will access and network services such as DNS and DHCP.

In addition, it typically deploys network management services, including monitoring, logging, and troubleshooting such as RADIUS, Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Syslog traffic.

2 Faults, failures, and symptoms

In the field of fault management, there are three basic concepts: faults, error, and symptoms. Faults (also referred to as problems or root causes) constitute a class of network events that can cause other events but are not themselves caused by other events. Error is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition. As the Figure 1.2 shows, an error is a consequence of a fault. Faults may or may not cause one or more errors. Errors may cause deviation of a delivered service from the specified service, which is visible to the outside world. The term failure is used to denote this type of error. Finally, symptoms are external manifestations of failures (Steinder & Sethi, 2004a).



2.1 Types of network failures

This research has taken the most widely and commonly failures classification used, which defines two types of failures: link failures and device failures. A link failure occurs when there is an event causing the connection between two devices to be down or excessive packet discards in a link. Usually, such failures are detected by SNMP monitoring on the interfaces of the devices. A device failure occurs when there is an event causing a device to be inoperable for routing/forwarding traffic. These failures are produced when a device is crash due to hardware errors or powered down for maintenance (Potharaju & Jain, 2013).

Device failures are usually caused by maintenance tasks (Gill et al., 2011), so they have a predictable occurrence. Meanwhile, link failures are more frequent, variable, and bursty. Thus, link failures are the focus of this research.

As shown in Figure 1.3, we classify the link failures, on the one hand, according to their impact, and according to the cause on the other.

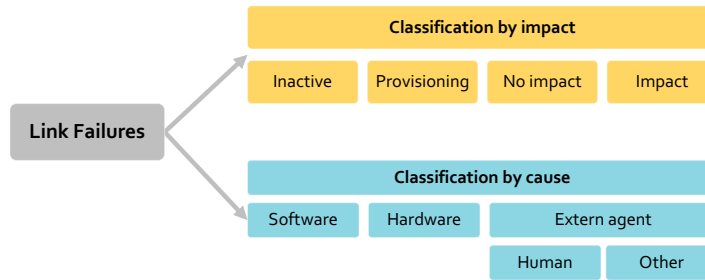


Figure 1.3 Link Failures types.

The classification by impact defines four link failures types based on traffic observations before and during failure. First, the inactive type occurs when there is no data before and during failure. Second, the provisioning type occurs when there is no data before failure and some data during failure. Third, a link failure with no impact occurs when there is about the same amount of data traffic before and during failure. Fourth, a link failure with impact occurs when the traffic significantly goes down during failure (Gill et al., 2011). Table 1.1 summarizes the traffic behavior for each link failure type.

Table 1.1 Traffic beffore and during link failure.

Link failure (by impact)	Traffic before failure	Traffic during failure
Inactive	no data	no data
Provisioning	no data	some data
No impact	data	data
Impact	data	data goes down

The classification by cause discriminate the link failures in those caused by software errors, by hardware errors, and by an external agent.

2.2 Fault diagnosis

Fault management is a multi-stage process that includes two activities: fault diagnosis and fault recovery (see Figure 1.4). Fault diagnosis is addressed by three steps: fault detection, fault localization, and testing. Fault detection consists of network symptoms observation in order to determine if one or more failures have occurred. Fault localization deduces the location of the failures in a network by using the set of observed symptoms. Testing checks the inferred failures. After the fault has been identified, the fault recovery process fixes it (Dusia & Sethi, 2016b).

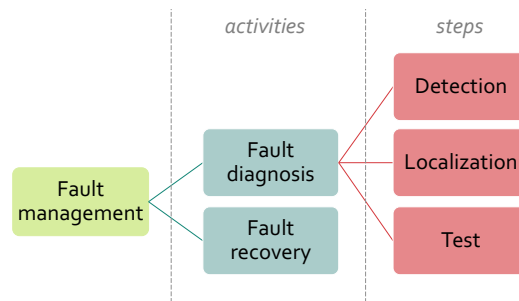


Figure 1.4 Fault management process.

3 Data stream

A data stream is a sequence of data that arrive ordered and separated by time intervals. It is a sequence $\langle S_1, S_2, \dots, S_n, \dots \rangle$ where each S_j is a set of instances (unit set for online learning). Each instance is generated according to a distribution D_j (Fernández et al., 2018a). Some examples include credit card transactions, sensor measurement, telecommunication, and more.

As (Fernández et al., 2018a) note, the following differentiates data streams from traditional datasets:

- Each data item in the data stream arrives sequentially over time.
- We do not have control over the data order, so it is compulsory to be ready to react to them at any time.
- Data streams are potentially infinite, so it is almost impossible to save all of their items in memory. The previous generally implies that a data item is only processed once, then it is discarded, and sometimes is stored if it is indispensable.
- Data streams are susceptible to changes; in other words, the data can change their distribution over time.
- Data labeling is costly and sometimes impossible.

There are two types of data streams: stationary and evolving. Stationary data streams are those relating to historical data or with a regular bulk arrival (Patil, 2019). The instances of each set S_j have a distribution D_j , and this distribution is unchanged for all instances set of a stationary data stream. That is, the distribution D_{j+1} of a set S_{j+1} is equivalent to D_j distribution ($D_j = D_{j+1}$) (Fernández et al., 2018a). Meanwhile, evolving data streams refer to real-time data, so it updates continuously. It means that the distributions D_j always change over time, so $D_j \neq D_{j+1}$ (Fernández et al., 2018a; Z. Li et al., 2020). This phenomenon is known as concept-drift. We can categorize concept-drift in two ways: according to the influence on learned decisions or classification boundaries, and according to the speed of changes taking place within the stream.

In the first branch, the concept-drift is virtual or real. Figure 1.5 illustrates both. When a virtual drift occurs, the change in the data distribution over time does not affect the probability that an instance will be classified as one class or another. So, the classification boundaries will remain consistent over time. Meanwhile, when a real drift occurs, said data distribution change affects the probability that an instance belongs to a class. This means that the classification boundaries will no longer become effective, so the learning model must change (Priya & Uthra, 2020).

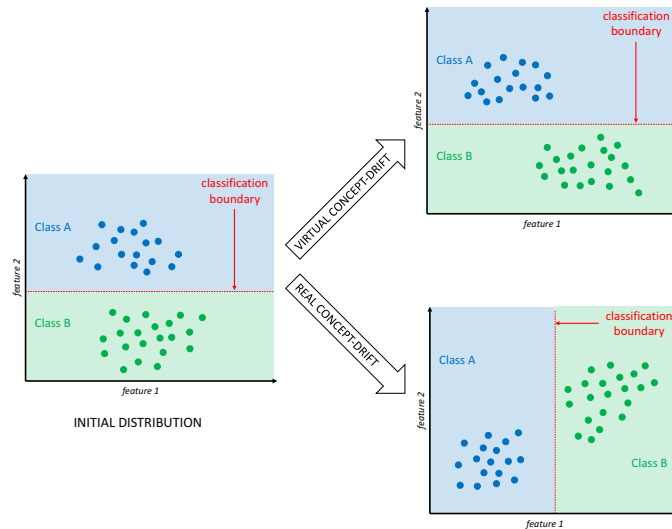


Figure 1.5 Concept-drift types according to the influence on learned decisions or classification. Based on the image provided by (Fernández et al., 2018a).

In the second branch, concept-drift is classified as sudden, gradual, incremental, or recurring. Figure 1.6 illustrates each. A sudden concept-drift causes a sudden and significantly different change in the data distribution. It is also known as a shift (Bifet et al., 2018). When gradual and incremental concept-drift occurs, data distribution undergoes slight and continuous changes, so the accumulated changes become significant after a while. The difference between gradual and incremental drift is based on the ratio of changes (incremental drift has a slower ratio of slight changes). Recurring concept-drift refers to the reappearance of past distributions (Priya & Uthra, 2020).

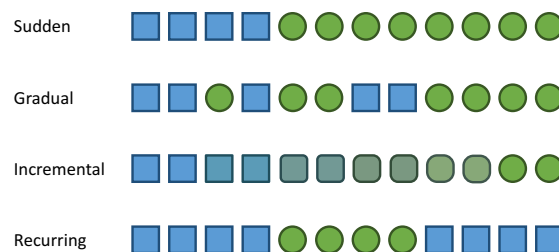


Figure 1.6 Concept-drift types according to speed of changes taking place within the stream. Based on the image provided by (Fernández et al., 2018a).

It is essential to highlight that in real-world applications, generally, there are combinations of several kinds of concept-drift.

4 Machine learning

Machine Learning (ML) is an artificial intelligence field that tackles the idea of systems learn from data and identify patterns for making predictions on new data without explicit programming instructions or human intervention. So, ML explores the study and construction of algorithms that can learn and predict. Those algorithms learn from experience, that is, existing data (existing instances). The algorithms discover data patterns from a data set (examples of data), resulting in a

model; it is a process denominated as training. The model is a mathematical function representing the learned patterns (generally complex depending on the algorithm and the data) and allows to predict on new data (new instances) not seen in training (Witten et al., 2017).

For information, the following are the traditional ML algorithms used in experiments.

- **Naive Bayes:** It is a machine learning algorithm for classification tasks, which uses the Bayes Theorem and assumes that all the predictive attributes are independent of each other. The independence principle is not always fulfilled in reality; nevertheless, Naive Bayes is a very efficient algorithm, even if this criterion is not met (Webb, 2010).

As (John & Langley, 1995) explains it, let Y the class variable (target variable) and X be the vector of the observed predictive attributes values. Suppose y is a particular class and x is an observed attribute vector to classify (new instance). In that case, the probability of each class must be calculated given the vector x (using (1) formula) and select the class with the highest probability.

$$p(X = x | Y = y) = \prod_i p(X_i = x_i | Y = y) \quad (1)$$

- **Multi-Layer Perceptron (MLP):** The artificial neural networks are based on the biological idea of the brain neural networks: the neurons are connected to propagate electrical signals through the network (send and receive signals between them). The neurons process input signals and can be activated if the electrical input crosses a threshold. Then, that activation allows them to send the signal to another neuron.

Applying the above concept, the artificial neural networks have the structure shown in Figure 1.7. The units represent the neurons, and weighted edges connect them. The units are arranged into at least two layers: the input and output layers. The input layer stores the data to be processed (the predictors or attributes), and the output layer gives the results (the target variable). If there are hidden layers between the output and input layers, the artificial neural network is called Multi-Layer Perceptron.

To make a prediction, the units in the first hidden layer receive a weighted value (w_i) of each input unit (x_i) and, considering a bias (w_0), computes the activation function (f) on the values to get an output. The output value is weighted and propagated to the next layer. This process is repeated until the output layer is reached. The hidden layers use the same activation function, while the output layer generally uses a different function.

Given a set of input training data, the activation function to use, and the number of hidden layers, MLP finds the best bias (w_0) and weights (w_i) to perform predictions.

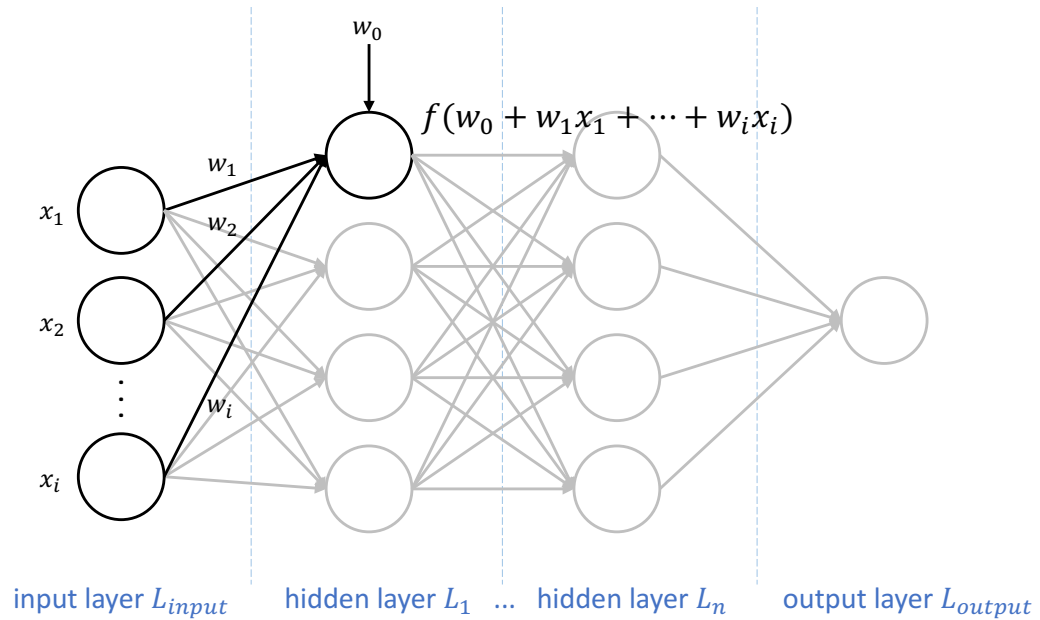


Figure 1.7 Multi-Layer Perceptron Representation.

- **K-Nearest Neighbors (KNN):** The KNN algorithm is introduced by (Cover & Hart, 1967), and it does not have a training process as such; instead, it performs an instance-based classification, which is why it is referred to as a lazy algorithm. Given a new instance to be classified, KNN measures the similarity in terms of distance between the new and existing instances to select the closest k neighbors. The assigned class corresponds to the majority class of the found nearest neighbors (Witten et al., 2017).
- **AdaBoost:** It is a meta-algorithm that embeds any weak machine learning algorithm to boost the performance using a sequential learning technique.

Figure 1.8 shows the training process of AdaBoost using any algorithm. The training is a loop starting with all training instances having equal weights. For each cycle, the weak learning algorithm uses data weighted for training and produces a weak model. Then, AdaBoost updates all the weights on all the training instances according to the residual error of the previous weak model, so weights are not going to be equal at the next training cycle. In this way, AdaBoost gives greater weight to those misclassified observations in the previous model. The loop continues until a certain number of weak models have been created (preset by the user). For making predictions, given a new instance to be classified, AdaBoost averages the models' predictions (Freund & Schapire, 1996).

The AdaBoostM1 algorithm version (Freund & Schapire, 1996) is used for experimentation issues in this work.

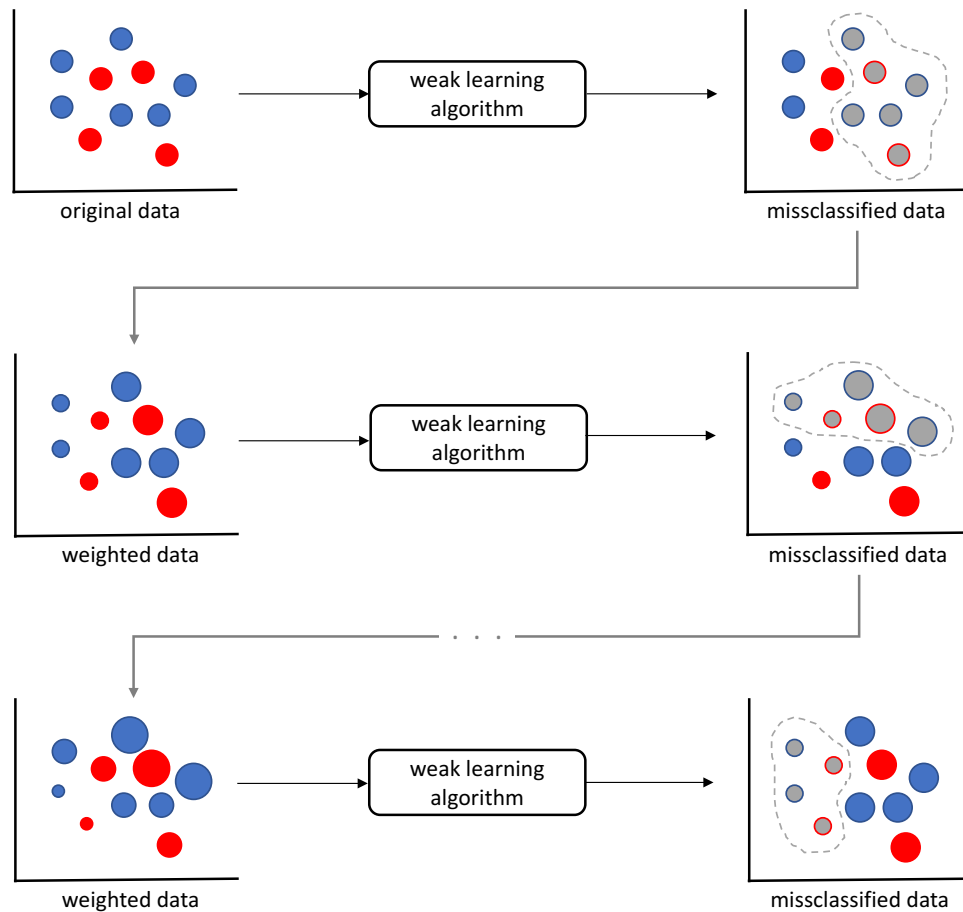


Figure 1.8 AdaBoost algorithm.

- **Bagging:** Bootstrap Aggregation or Bagging is a meta-algorithm that uses a parallel learning technique and any machine learning algorithm (a base learner). Several training subsets are extracted randomly given a training dataset, and then Bagging fits the base learner on each subset separately. The number of created models is equal to the number of sets. For the prediction task, Bagging averages the models' predictions to provide a prediction. (Breiman, 1996).
- **Decision Tree:** The algorithms that create a model looking like a flowchart are referred to as decision trees, so humans can easily understand those models. As Figure 1.9 shows, each diagram level represents a decision question related to an attribute with two options as answers. Following a sequence of questions and answers, the model finds a classification for a new instance. The model has four components: the internal nodes representing the attributes, the branches representing a decision rule, each leaf representing the result, and a root node at the top of the tree.

The algorithm selects the best attribute using an attribute selection measure in the training or tree creation process. Then, it sets that attribute as the root node, which divides the data into smaller subsets. This procedure is recursively repeated for each attribute until there are no more cases, there are no more attributes, or all variables belong to the same attribute value.

In this work, the C4.5 decision tree (Salzberg, 1994) is used for experimentation issues.

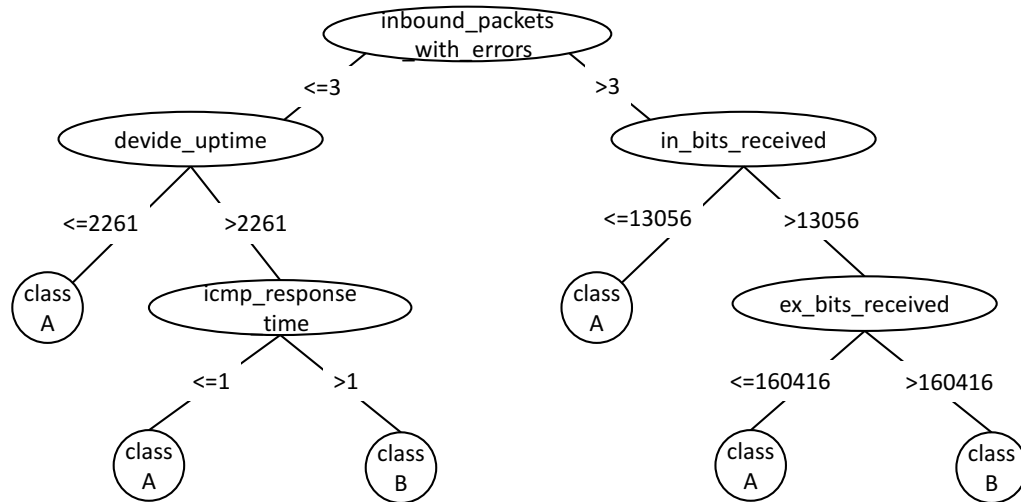


Figure 1.9 Example of a decisión tree.

- **PART:** It is a rule algorithm; consequently, its function is to create a decision list as its model. It iteratively builds partial C4.5 decision trees (one tree for each iteration) and selects the best leaf as one of its decision rules (one rule for each tree) (Frank & Witten, 1998).

5 Incremental learning

Traditional approaches to knowledge extraction assume that all the data needed to generate an inference have been collected and are available in a training set (Bramer, 2013). However, the more data available, the more performance difficulties of machine learning algorithms will have. Also, there are more and more situations in which there is a large volume of information to extract knowledge from it (Del Campo-Ávila, 2007). Hence, traditional approaches have limitations to process large amounts of data, especially data continuously growing. So, incremental learning algorithms arise to solve these and other issues.

Algorithms with incremental learning (also known as online learning) are those that can update their learning model using new arrival data without re-processing the data already used. Then, they discard the training data if it is no longer required. These algorithms are used when the data to process are not all available at the beginning of the process (Bramer, 2013). Also, they are ideal when the input to a learning process is streaming data, in other words, when data that arrives in real-time as an infinite stream making predictions in real-time while refining the model according to the changes in the evolving input stream (Utgoff, 2017; Witten et al., 2017).

Two fundamental aspects characterize incremental learning. Firstly, it is capable of incorporating information from new experiences, which were not previously available in the dataset, to the model. Secondly, it is capable of making the model evolve so that it increasingly represents more complex concepts. So, the term "any-time learning" emerges (Del Campo-Ávila, 2007).

As will be seen, this work addresses the fault diagnosis issue in an IP-based network using machine learning techniques, more specifically by incremental learning approaches which process the monitoring data of the network that are constantly being generated as a data stream. In this chapter, we introduced the fundamental knowledge relevant to this research, so the vocabulary used in the further sections is based on the previous literature for a better understanding.

Chapter II: State of the Art

*“¿Quién les dio la verdad absoluta?
Nada hay absoluto. Todo cambia, todo se mueve,
todo revoluciona, todo vuela y se va...”
(Frida Kalho)*

1 Network fault diagnosis through years

A bibliometric study through science mapping around network fault diagnosis offers an overview of this topic through years. We used the open-source science mapping software tool called SciMAT (Cobo et al., 2012) for this purpose. This section describes in-depth the followed steps to build up the state of the art study.

1.1 Methodology

The methodological foundation of SciMAT is the general workflow in a science mapping analysis. So, it incorporates the needed modules to perform the steps of this mapping workflow and enables us to follow it flexibly. Figure 2.1 shows the workflow configuration according to our purpose. Each step is described below.

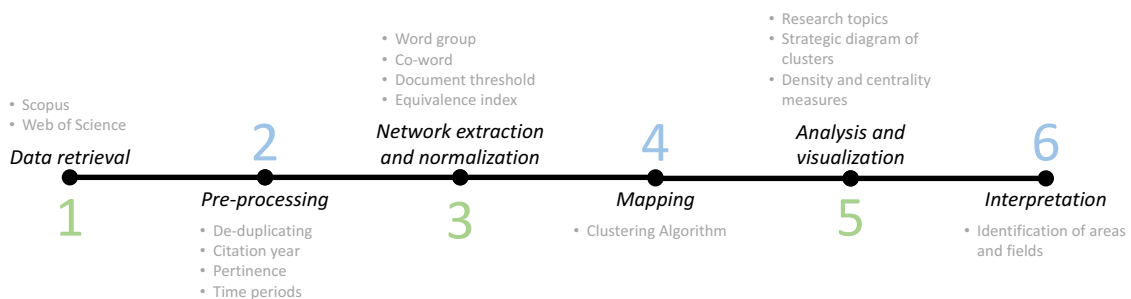


Figure 2.1 Workflow of bibliometric analysis with SciMAT.

1. Data retrieval:

We gathered the relevant work about fault diagnosis in networking from scientific bibliographic databases Scopus and Web of Science (WoS). The used descriptive terms were fault diagnosis (including synonyms: fault detection, failure detection, and failure diagnosis), network management

(including synonym: network monitoring), network (including synonyms: computer network, IP based network, and communication network). Also, the following terms were excluded to guarantee a strong concerned scientific field description: electrical network, nano network, sensor network, and power network.

The retrieved number of works was 1720 from Scopus and 220 from WoS, from 1990 until now. 1940 documents in all.

2. Pre-processing:

Preprocessing allows filtering the set of related works found to refine bibliometric study. Using the SciMAT tool, this procedure consisted of delete duplicate items and not relevant (because the search yielded some works outside the area of interest). The period between 2001 and today was selected for the study. The data was divided into four subperiods in order to analyze the temporal evolution of the scientific field (2001-2005, 2006-2010, 2011-2015, 2016-2020). As a result, the corpus of information was reduced to a size of 1535 documents.

3. Network extraction and normalization:

This step makes possible the construction of the bibliometric network, that is, to detect all relationships between documents. For this purpose, we set the keywords present in the scientific papers as the unit of analysis. So, a word grouping process was performed manually so that the SciMAT tool could recognize associated terms or expressions. Then, the SciMAT tool was configured to build a bibliometric network from the co-occurrence of keywords, thus generating a co-word network.

Also, to keep only the most significant works in the resulting network, a minimum reference frequency was configured (three references is our threshold), and the equivalence index as the similarity measure for normalization. Equivalence index is a measure that takes into account the number of co-occurrences of the words in a set of documents, and the number of documents described by those words, so it enhances the pairs of keywords that adequately represent the corpus analyzed.

4. Mapping:

SciMAT locates the groups of keywords closely linked to each other. Those groups correspond to the focuses of interest or research problems in which researchers have been concentrated. The tool executes this process using a clustering algorithm (Simple Centers Algorithm (Cobo et al., 2011; Coulter et al., 1998)).

5. Analysis and visualization:

In this step, SciMAT takes quality measures (h-index and average citations) into consideration to performs the analysis. The tool deploys the following elements:

- Evolution maps: it allows us to see each period's clusters and their relationship with other neighboring periods in a single graph. It is known as longitudinal view.

- Strategic diagrams: it is a two-dimensional space built that places the topics (clusters) in a quadrant according to their type. Four topic types are considered: motor, highly developed and isolate, emerging or declining, and basics and transversal.
- Clusters' network: they highlight the relationship between keywords into a detected cluster. One graph by each cluster.
- Documents associated with the cluster: List of highest impact documents related to the cluster.

SciMAT builds all listed graphics. As a result, 61 prominent clusters or topics were obtained in the research field during the entire observation window (2001 to 2020).

6. Interpretation:

Figure 2.2 shows the appearance of an evolution map. Each circle represents a cluster or detected topic. Those topics are closely related to one or more periods, so one cluster can appear in more than one column. Each column represents a period, and they are ordered in time. The lines represent the relationship between topics. The thicker and repainted the line, the closer the relationship is.

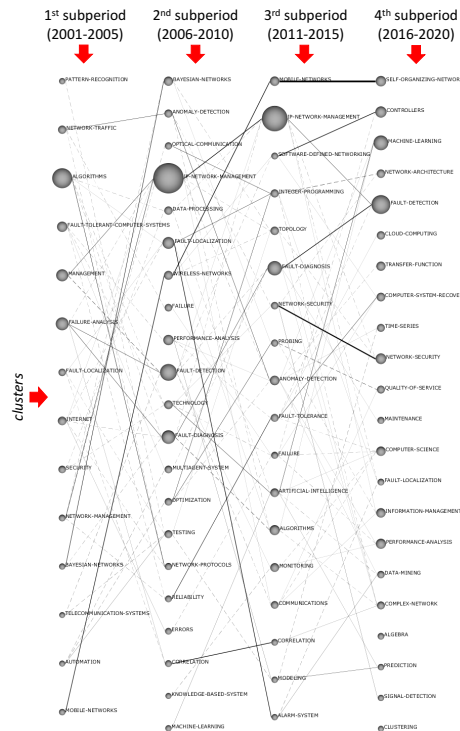


Figure 2.2 Evolution map appearance.

Figure 2.3 is a Cartesian plane that indicates how SciMAT distributes the detected topics according to their types. In the upper-right quadrant are well-developed and essential topics for the construction of the research field. These topics are known as *motor clusters*.

The upper-left quadrant collects very specialized topics, so they have very well developed internal links, but their importance is marginal in the research field. These topics are known as *highly developed and isolate clusters*.

Very underdeveloped and marginal themes are located in the lower-left quadrant. It means that they mainly represent appearing or disappearing topics, so their name is *emerging or declining clusters*.

The topics in the lower-right quadrant are known as *basic and transversal clusters*. They are essential to the scientific field but are not well developed. Those clusters are transversal and generic topics, so they are the basic topics of the scientific field.

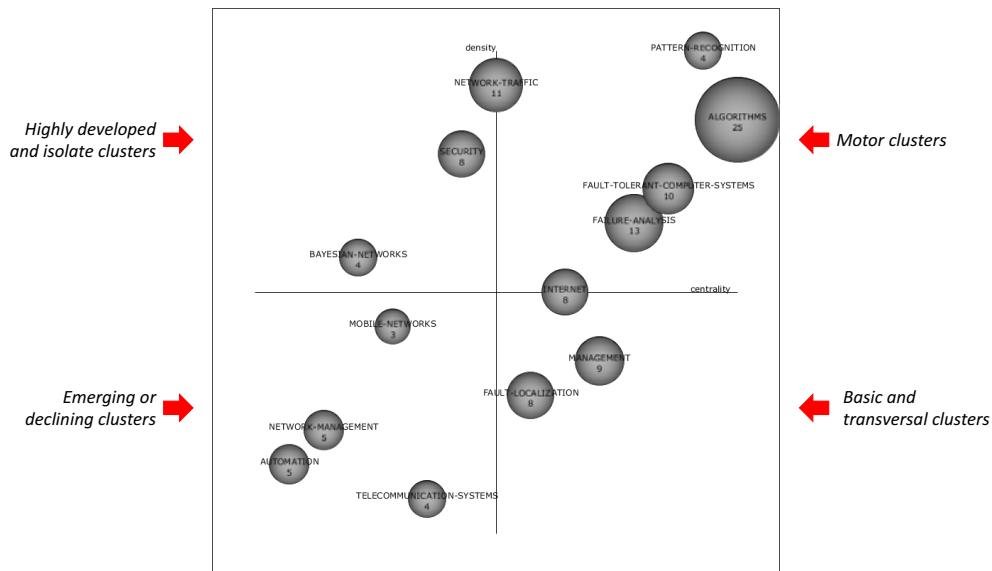


Figure 2.3 Strategic diagram appearance.

Figure 2.4 is a cluster network example. As shown, a network is a set of interlinked words, so no cluster is isolated.

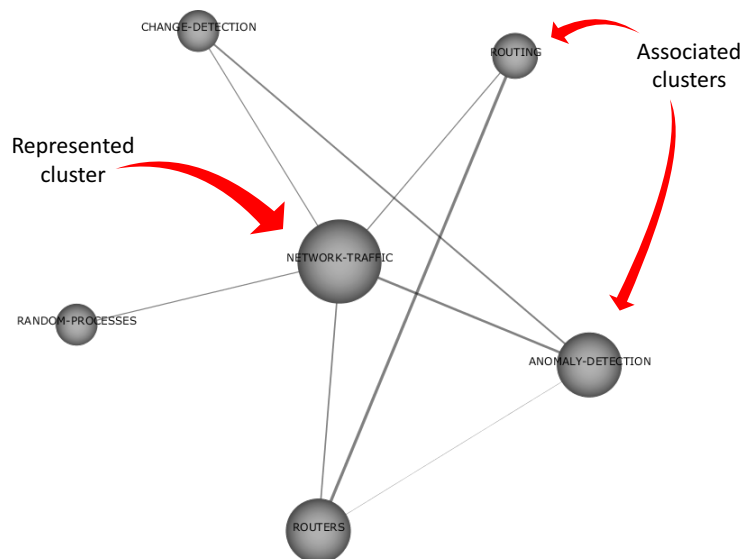


Figure 2.4 Cluster network appearance.

The next section contains a more in-depth analysis of the results obtained with the SciMAT tool.

1.2 Analysis of results

To obtain a scientific panorama over the years, we have analyzed the research and trends of developments around influential topics in diagnosing failures in communications networks for every subperiods. Those topics correspond with the clusters found by SciMAT. We heeded those clusters that stand out in various subperiods of time to observe the research focus's evolution deeply.

1.3 First period (2001-2005)

As we can see in Figure 2.5, the cluster “algorithm” contains the principal motor works and repeats its appearance in another period, as we will see later. Its size is because it collects works from other spheres in the same period, which base their research on algorithms. Going into this cluster, we found that the traffic analysis process led to the studies (Krishnamurthy et al., 2003). This process was done in order to see traffic changes as an indication of anomalies. Signal processing and log analysis are two other motors processes of fault diagnosis. The first approach was envisioned as a great potential to enhance the anomaly detection field and improve IP networks' reliability (Steinder & Sethi, 2004a). The second, to detect frequent patterns from logs and to identify anomalous log file lines (Vaarandi, 2003).

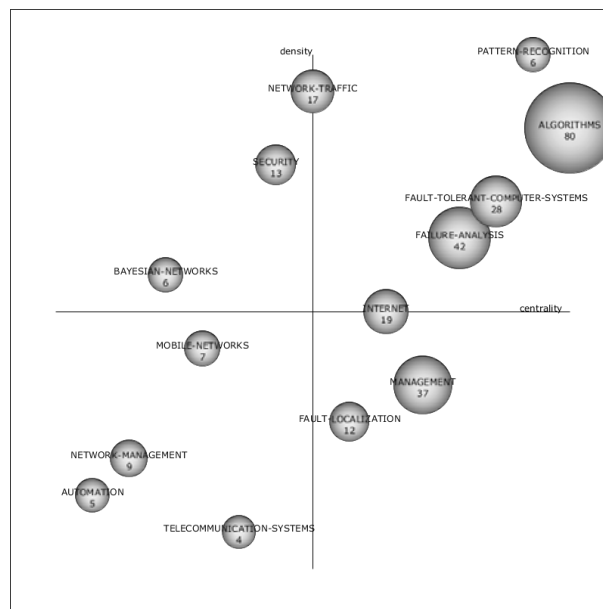


Figure 2.5 Strategic map for period 2001-2005.

Exploring the lower-right quadrant works, or basic and transversal clusters, especially those of the “fault-localization” cluster, it is evident that the fundamentals of the topic are bayesian reasoning (Steinder & Sethi, 2004c), probability, and event-driven (Steinder & Sethi, 2004b).

On the other hand, as we can see in the upper-left quadrant, “security” is an isolated cluster, but “bayesian-networks” is a highly developed topic because of its works. Those works are concerned with improving the probing methods for active monitoring using mainly Bayesian networks (Kirmani & Hood, 2004; Rish et al., 2005).

On the other hand, it is interesting to see through the lower-left quartile that the “mobile-networks” cluster emerges. It was precisely in this period that the 2.5G technology for voice and data support arose. The popularization of VoIP also occurred in 2003, so works to diagnose faults in those networks emerged (Ritter et al., 2004; Xueshan Shan & Li, 2001).

1.4 Second period (2006-2010)

Figure 2.6 shows the strategic map for this period. Now, it is evident the clusters increment, and there are new topics around fault diagnosis.

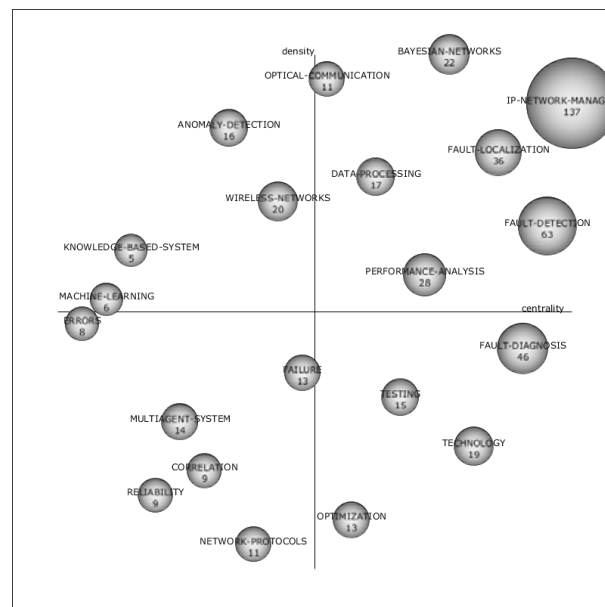


Figure 2.6 Strategic map for period 2006-2010.

Let us look at the upper-right quadrant first. All motor clusters of this period incorporate active probing for network monitoring and combine it with other approaches for different network diagnostic purposes (such as studying fault propagation, detecting congestion, and detecting performance problems) (L. Cheng et al., 2010; Natu & Sethi, 2006).

It should be noted that within the motor clusters, there are two that were also in the previous period, but in a different quadrant. The cluster "fault-localization", a basic transversal during the last period, is now a motor topic that incorporates work based on association rules, alarm correlation, and fault propagation (Huang et al., 2006; T. Li & Li, 2010). The "bayesian-networks" cluster was highly-developed in the previous period. It now uses all the knowledge of Bayesian networks in broader studies, such as avoiding congestion, modeling fault propagation, fault location, and agents' use (Cho et al., 2008; García-Algarra et al., 2011; Jian, 2010).

The "network-management" cluster classified as emerging in the previous period made way for the motor sphere "IP-network-management". The works under this cluster are mainly focused on studying congestion and performance problems in optical (Harvey et al., 2007), IMS (Reali & Monacelli, 2009), and IPTV networks (Mahimkar et al., 2009).

Now let us look at the lower-right quadrant to learn about the transversal themes of the period. If we study the quadrant's works in-depth, we will see that the basic theme revolves around the temporal analysis. Researches find spatio-temporal patterns through temporal correlation (Natu & Sethi, 2008a; T. Wang et al., 2010). Here also works focused on overlay networks are highlighted (Natu & Sethi, 2008b).

In the upper-left quadrant, two clusters stand out, "anomaly-detection" and "machine-learning". We will consider the first topic a developed theme as it focuses its efforts on the study of traffic to detect anomalies (Chhabra et al., 2008), and as we saw in the period 2001 to 2005, this approach was a driving theme. While the second cluster, we consider it isolated, as it is dispersed among SVM (Support Vector Machine) topics (Guo Jiangwei et al., 2010), digraphs (C. Li et al., 2009), adaptive probes (L. Cheng et al., 2010), and exploring AI as promising for network management (Qi et al., 2007).

Finally, in the lower-left quadrant, we see emerging topics such as the study of fault resilience in metro ethernet networks (Huynh et al., 2007), specific studies for link failures (Fraiwan & Manimaran, 2008), and event correlation implementation (Hanemann, 2006).

1.5 Third period (2011-2015)

As Figure 2.7 shows, the upper-right quadrant includes motor topics that, for the most part, had already appeared in previous periods but different quadrants. The "mobile-networks" cluster was an emerging topic in the first study period (2001-2005). The "fault-diagnosis" cluster between 2006 and 2010 was a transversal topic. The "IP-network-management" cluster is also an emerging sphere since the previous period. Scrutinizing through these recurring clusters' research work, we discover that the period's motor is the study of anomaly detection in LTE mobile networks applying several approaches.

It is important to note that the introduction of LTE technology in the world takes place in these five years. So, we consider this to be the cause of the intense research focus on these networks. Some noteworthy examples of motor works belonging to the three clusters mentioned above are (Asghar et al., 2012; Nováczki, 2013; Szilagyi & Novaczki, 2012).

Another motor topic of the period is "artificial-intelligence" which focuses on applying this scientific field to provide support to make decisions against attacks (Arendt et al., 2015). The cluster also uses artificial intelligence to build systems network triage alarms to help operators find and fix problems (Amershi et al., 2011), correlate alerts (Salah et al., 2013), and the study of failures from the user's perspective (Takeshita et al., 2015).

The lower-right quadrant has the basic and transversal topics where there are also recurring clusters, such as "algorithms" and "anomaly-detection." Within these clusters, we mainly find different approaches for detecting anomalies in large-scale IP networks (He et al., 2012; Kanda et al., 2013).

Let us move to the upper-left quadrant. It contains highly developed clusters oriented to processing complex events (S. Cheng et al., 2011), the study of failures in multilayer and overlay networks

(Steinert et al., 2011), diagnosis of fiber failures (Yuan, C. et al., 2015), and detection of attacks, especially DDoS type (F. Wang et al., 2012).

The cluster "software-defined-networking" also stands out, between the limit of research classified as emerging and those classified as isolated. It is an emerging topic because works studying faults in SDN with simulated experiments are beginning to appear (Adrichem et al., 2014; Kamisiński & Fung, 2015; Sharma et al., 2013). However, it is also isolated because all the studies of the period around SDN networks do not fall precisely on fault diagnosis.

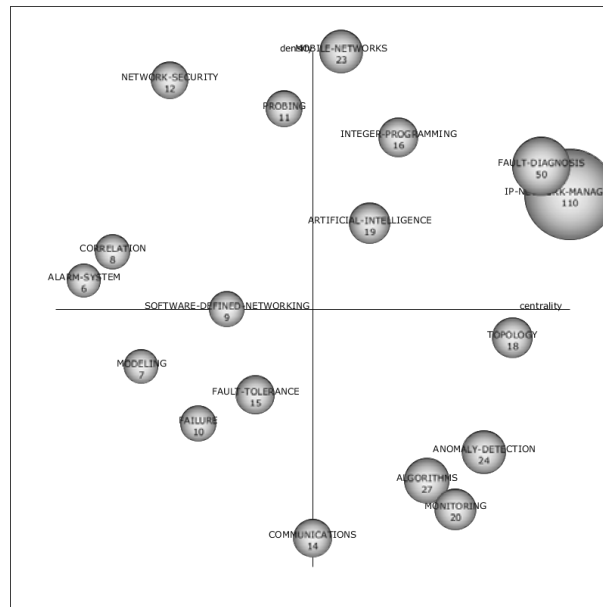


Figure 2.7 Strategic map for period 2011-2015.

Finally, according to the lower-left quadrant, works focused on strategies to mitigate failures (X. Wu et al., 2012), and natural language processing to analyze trouble tickets in a monitored network emerge (Potharaju et al., 2013).

As could be seen, in this period, there are several clusters of previous periods, but located in a different quadrant, this is a sign of the dynamism of our research area.

1.6 Fourth period (2016-2020)

This last period is remarkable. If we take a general look at all the works analyzed during this period, we will realize that the vast majority are related to SDN networks, regardless of the quadrant to which they belong (Bu et al., 2016; Tang et al., 2016). The above makes sense because, over the last few years, SDN networks have not only represented a network evolution but a disruptive technology. Let us remember that in the previous period, SDN was an emerged issue.

There are three recurring topics in the upper-right quadrant, of which the "machine-learning" cluster stands out, which was an isolated sphere between 2006 and 2010 and now is a motor topic. It is the motor of various studies related to the diagnosis of faults as identification of attacks and

malware (Watson et al., 2016), detecting abnormal flows (Kasai et al., 2016), and forwarding faults (Bu et al., 2016).

On the other hand, analyzing each sphere of the lower right quadrant, the basic and transversal topics, there are several related to machine learning and big data, such as (Lin et al., 2016; Musumeci et al., 2019; Yang et al., 2017), regardless of the sphere to which they belong. Some of these works use these techniques for data stream analysis (Ahmad et al., 2017). An in-depth analysis shows us that the research in this quadrant focus on the study of network performance.

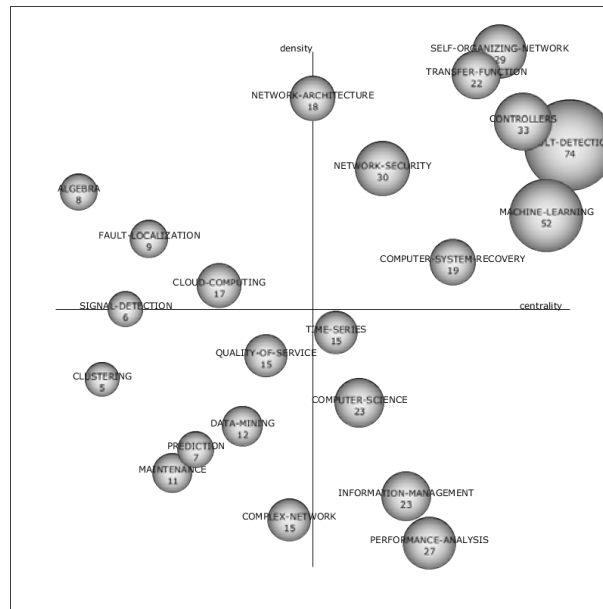


Figure 2.8 Strategic map for period 20016-2020.

The transition of services to the cloud or cloud computing has become a fundamental part of many companies' business strategy in recent years. As shown in the upper-left quadrant, the "cloud-computing" cluster appears as an isolated topic, but its proximity to the emerging research quadrant is also evident. This sphere contains works mainly oriented to the analysis of anomalies in cloud environments (Colman-Meixner et al., 2016; Sauvanaud et al., 2018; Watson et al., 2016).

Finally, it is interesting to analyze the works in the lower-left quadrant, where there are clusters such as "quality-of-service", "prediction", and "data-mining". Network management tasks tend to improve the quality of service for users during this period (Ahmed et al., 2016; H. Wang et al., 2017). Most significantly, prediction tasks are closely related to the emerging data mining process in the last five years within the fault diagnosis field (Duenas et al., 2018; Ozelik & Yilmaz, 2017).

As we have appreciated, network fault diagnosis has been a constant research topic, that has evolved according to the networks' evolution and their new needs over the years. So, much still remains to be done and discuss.

2 Network fault diagnosis techniques

We can extract the six approaches used in fault diagnosis tasks shown in Figure 2.9 from the periods studied.

Passive techniques monitor a network by deploying monitoring agents on networking devices. So, any failure condition in the network could generate multiple alarms by monitoring agents. These techniques are named passive because the Network Management System waits passively for alarms to be sent by the agents, and then use as a symptom to analyze the exact failure condition in the network. (Steinder & Sethi, 2004a) is a comprehensive survey of these approaches.

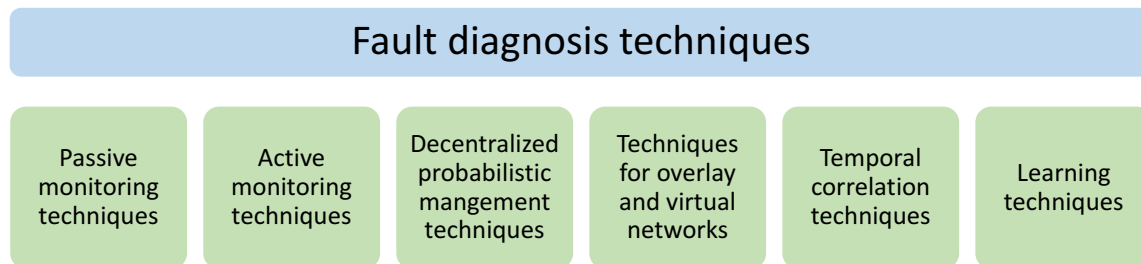


Figure 2.9 Fault diagnosis techniques.

Active monitoring techniques use probing for a variety of network management applications. A probing station is a node in the network that transmits one or more packets called probes to monitor the state of the network. Examples of probes may be ping or traceroute; probes may also be more complex and may be handled by any protocol layer. The use of probes to determine the network behavior or measure the quality of network performance is called probing (Dusia & Sethi, 2016a). Active monitoring techniques are still being used today. Two examples of their use in the current decade are (Likun Yu et al., 2010) and (Lu et al., 2013), which use probing for a variety of network management applications.

The decentralized probabilistic management approaches budded with the goal of decrease the traffic load to a central management node. In this technique, each network node has embedded a management process. The network nodes communicate with each other, and the final analysis is reported to the network administrator. Some examples of related works that propose this type of technique are (Steinert & Gillblad, 2010) and (Prieto et al., 2011).

Around 2007, Software-Defined Networks (SDN) emerge and, therefore, the techniques for overlay and virtual networks, such as those proposed by (Gillani et al., 2014; Yan et al., 2014) and (H. Wang et al., 2015). These approaches focus on solving different challenges of fault diagnosis process for this type of networks, such as inaccessible substrate network fault information, incomplete and inaccurate network observations, dynamic symptom-fault causal relationships, and multilayer complexity.

The temporal correlation techniques like (Z. Li et al., 2009) and (Steinert et al., 2011) comes up in tackling the dynamic and multilayer nature of networks and correlate internal events for fault localization.

While the learning approaches, such as statistical and machine learning, have been adopted by other techniques over time to serve their intended purposes, they can be classified in a separate category. Then, if we separate the learning techniques as a specific type, these can be defined as those that diagnose the empirical data provided by the network management system. These techniques would not require a complete dependency model of a network. Some examples of related works that propose this type of technique are (Mahimkar et al., 2009) and (Kavulya et al., 2012), which use statistical approaches, and (Johnsson & Meirosu, 2013) and (Johnsson et al., 2014), which implement traditional machine learning approaches.

Each technique can favor or disfavor aspects of the network, as well as being a problem by itself. As proof of this, passive, active, and decentralized methods are the most common traditional techniques for fault localization, but they are invasive because they increase network traffic and control overhead. Techniques for overlay and virtual networks and the decentralized management increase the internal processes of the network because the first needs to install monitoring agents on all overlay nodes, and the last requires having an embedded management process on all networking devices. Temporal techniques mean analysis complexity, and all the methods named above need in-depth knowledge of the network connectivity and operations in addition to an extensive understanding of network behavior. So, fault diagnosis based on learning techniques can be useful because they would not require a complete dependency model of a network. Nevertheless, existing learning techniques to date, deal with insufficient data samples, and lengthy retraining may be necessary whenever the system behavior changes significantly.

3 Current fault diagnosis gaps

Each studied period has problems to face, but undoubtedly, some issues remain intact over time despite the techniques' evolution. We identify several gaps:

- a) None of the studied techniques is resilient to the network topology changes. For example, if network structure changes, the learning techniques will need to learn new symptoms-fault relationships.
- b) The existing learning techniques to date deal with insufficient data samples. They require lengthy retraining whenever the system behavior changes significantly because traditional machine learning techniques learn from static data, which will become obsolete because of that changes.
- c) A non-invasive method to continuously diagnose the network symptoms without leaving the system vulnerable to any failures has not been found in the state of the art revision. The existing models to date based on learning techniques must be thrown when the retraining period occurs, so the systems become vulnerable to any failures.
- d) A non-invasive approach resilient to the dynamic changes of the network, which can cause new failures with different symptoms, has not been found in the literature.
- e) Although there are studies that detect faults in real-time, a technique based on learning (non-invasive and without knowledge of network model) and real-time, has not been found.

4 Problem Statement

In brief, there is no a non-invasive technique to continuously diagnose network symptoms without leaving the system vulnerable to any failures, nor is there a technique resilient to the dynamic changes of the network which can cause new failures with different symptoms. Hence, this work focuses on to propose a fault detection approach that does not increase network traffic, control overhead, or internal network processes.

We hypothesized that if there is a phenomenon of failure propagation, the symptoms of failures can be observed at levels higher than their origin. The peripheral observation of symptoms guarantees a non-invasive network monitoring because neither the traffic nor the control overhead is incremented in the internal level of the monitored network.

This proposal also raises the combination of data analytics techniques based on incremental learning algorithms to process those failure symptoms peripherally observed. We conceive an online learning model for this purpose, that is, to process symptoms on-the-fly and continuously during the operation of the network to detect faults that occurred in internal network elements. This learning approach contributes to timely detect failures in an IP-based network. Additionally, it is resilient to dynamic network changes because the incremental learning models guarantee continuous learning from new symptoms adapting the model by self, while network change, so does the model.

Chapter III: PALADIN - A Peripheral Fault Diagnosis Model for IP-based Networks

“All sorts of things can happen when you’re open to new ideas and playing around with things.”
(Stephanie Kwolek)

This chapter presents the PALADIN model (PeripherAL fAult Diagnosis model for Ip-based Networks), representing this work's principal and general objective. This model proposes a module set to diagnose faults in IP-based networks through peripheral observation of failure symptoms, which are processed continuously and timely by incremental learning algorithms. In this way, a non-invasive and resilient fault detection to the dynamic changes of the network will be guaranteed.

PALADIN consists of four modules, as Figure 3.1 indicates. The **identify** module identifies the piece of the network in which make the diagnosis. The **collection** module sets the rules for monitoring parameters on the selected network piece. The **structuring** model establishes how to structure the monitored parameters before diagnosing faults. Finally, the **diagnosis** module proposes a learning model to detect failure from monitorisation results. The first module is executed only once, and it is the starting point of the entire model. All other modules are sequential, and their execution is continuous and infinite.

The following sections fully describe the proposed modules.

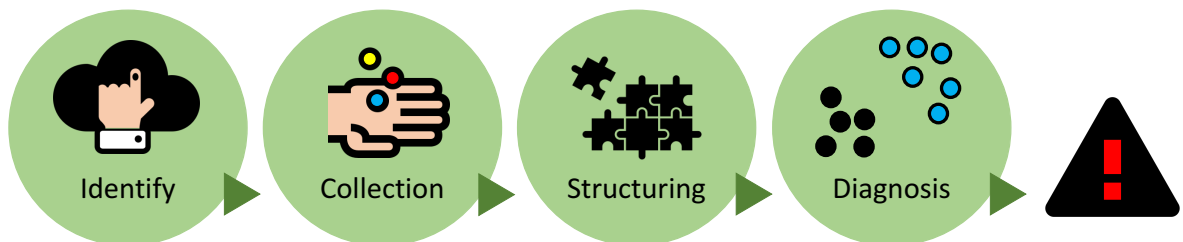


Figure 3.1 PALADIN - Fault diagnosis model for IP-based networks.

1 Identify

This work focuses on large IP-based networks with the industry wide adopted hierarchical model, as described in Chapter I. Then, before any monitoring network process, it is essential to determinate the hierarchical piece of network to diagnosis, named monitored network. For these monitored networks, we define the peripheral network elements as those that interconnect the network with other networks; in other words, are those that are closest to the Internet. Figure 3.2 shows an example of the concepts mentioned above.

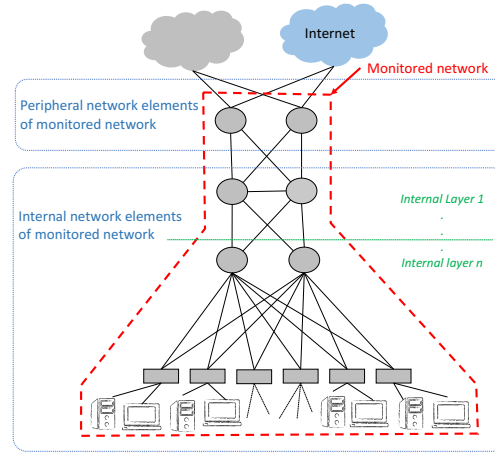


Figure 3.2 Elements of monitored network.

The fault diagnosis proposed by our model, detects failures caused in internal layers, through information from peripheral network elements only. These issues will be covered in-depth in the next sections.

2 Collection

There are several network data collection approaches, as shown in Figure 3.3 and surveyed by (Zhou et al., 2018). Our work focuses on collecting management information in addition to traffic information from the peripheral devices. So, the active probing mechanisms are the best option. We propose to use an *SNMP*-based collection mechanism to the parameter monitorization in the peripheral elements because the Simple Network Management Protocol (*SNMP*) has been widely adopted (Khan et al., 2018). Also, its use in a significant amount of management activities for enterprise and service provider networks by almost all networking vendors, make it popular (Narayanan et al., 2013).

This module defines a periodical collection of parameters polled on the peripheral network elements only. So, this approach decreases the monitorization traffic and control overload while networking management. Figure 3.4 shows the difference between traditional monitorization and our proposal.

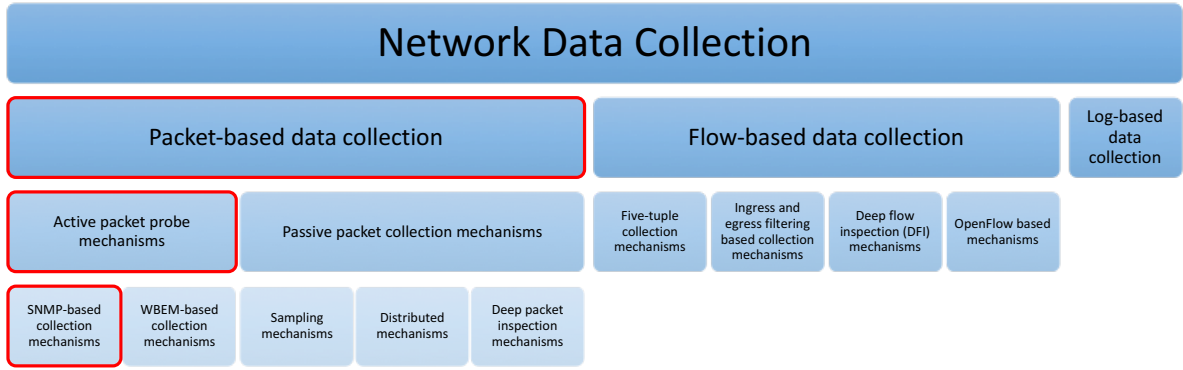


Figure 3.3 Classification of network data collection approaches, according to (Zhou et al., 2018).

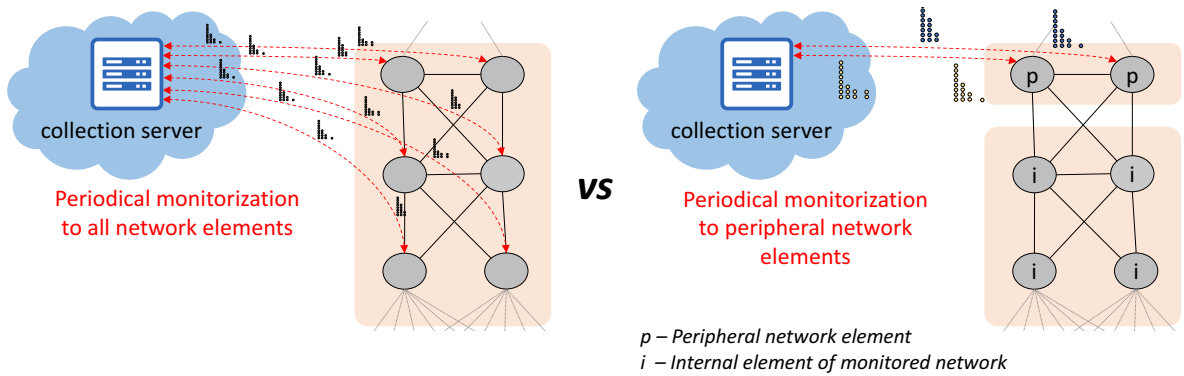


Figure 3.4 Traditional monitorization (left) vs peripheral monitorization (right).

Table 3.1 contains the parameters set to collect on peripheral elements. This set was selected from the analysis performed in the dataset construction described in Chapter IV. With respect to the collection or monitorization tool, any network management tool with *SNMP* support can be used (e.g., Zabbix (Zabbix LLC, 2021), Nagios (Nagios Enterprises. LLC, 2021), and Pandora FMS (Pandora FMS, 2020)).

Table 3.1 Parameters to collect.

PARAMETER	DESCRIPTION
Bits received	The total number of bits received on each interface of the device. It is measured for all peripheral device interfaces.
Bits sent	The total number of bits sent by each interface of the device. It is measured for all peripheral device interfaces.
Inbound packets discarded	The total number of inbound packets discarded by each interface of the device. It is measured for all peripheral device interfaces.
Inbound packets with errors	The total number of packets that contained errors, received on each interface of the device. It is measured for all peripheral device interfaces.
Operational status	The current operational state of each interface of the device. It is measured for all peripheral device interfaces.
Outbound packets discarded	The total number of packets discarded out of each interface of the device. It is measured for all peripheral device interfaces.
Outbound packets with errors	The total number of packets that could not be transmitted by each interface of the device because of errors. It is measured for all peripheral device interfaces.
Device uptime	The time since the device was last re-initialized.

SNMP availability	Peripheral device availability. Values: the device has not SNMP enabled, or the device has SNMP enabled.
ICMP response time	Time between echo request and echo response messages from SNMP server to the device.
ICMP loss	Percentage of lost packets.
ICMP ping	Device accessibility by ICMP ping. Values: ICMP ping fails, or ICMP ping successful.

3 Structuring

A parameter set is polled each period on the peripheral network elements. According to Figure 3.5, that means, there are k requests, one for each peripheral network element, so we expected k responses periodically (period T). However, as a result of the conditions specific to each network and the priority tasks of each device, one response is carried by several messages arriving in different instants; then, one response has a distribution of parameters over time. On the other hand, the PALADIN model should be independent of the configuration of the selected peripheral layer, where the number of interfaces of the monitored devices may vary, and therefore the amount of information that each response brings (the parameters of each interface). Under this scenario, the Structuring module has to provide m vectors of values i_i , where i_i (named instance) represents each polling response and m is potentially infinite, and each vector should have the same fixed length.

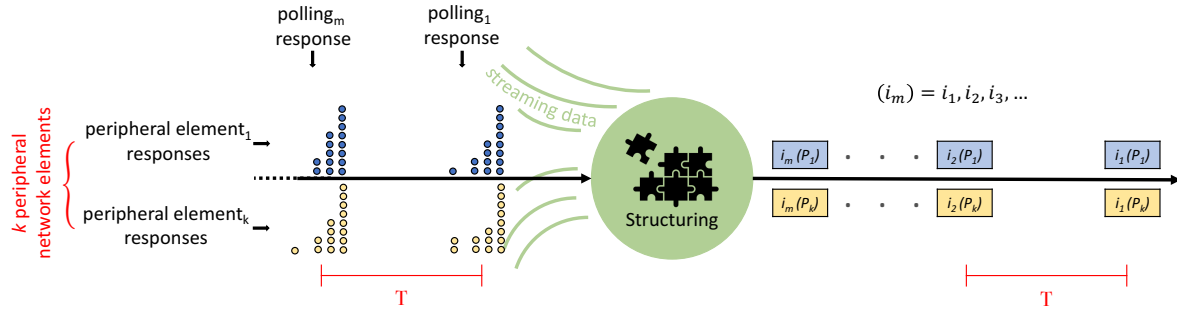


Figure 3.5 Structuring conditions.

Thus, the Structuring module has to deal with two issues to perform its mission successfully: several messages in response to one polling and several interfaces of the peripheral element. To solve them, we propose the two steps showed in Figure 3.6 for the structuring task. The first step is the grouping of messages that contain the parameters of the same polling. Section 2.1.1 of Chapter IV indicates this procedure in depth. The second step is grouping the parameters by three interface sets (Internal, Peripheral and External) to ensure a fixed structure for each response; that is, an interface-based grouping, as section 2.1.2 of Chapter IV describes.

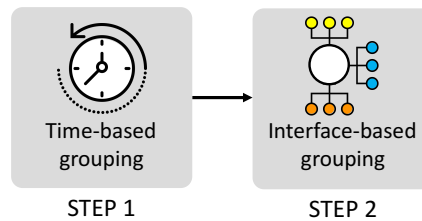


Figure 3.6 Structuring steps.

In this way, as the Structuring module result, each response has a fixed number of parameters and a shared timestamp. Its structure is the one raised for the instances of the SOFI dataset (Symptom-Fault relationship for IP-Network) described in Chapter IV. This data set construction follows the first three modules of the PALADIN model and constitutes the reaching of the first specific objective of the thesis.

4 Diagnosis

This process must consider that one instance of the SOFI dataset is obtained for each polling to a peripheral device (*k peripheral network elements*). Moreover, a label is assigned to each internal layer characterising the on failure or healthy network state, so the number of internal layers of the monitored network define the number of labels (*n*) the model must deal with; then, we face a multi-label learning problem as indicated in Figure 3.7.

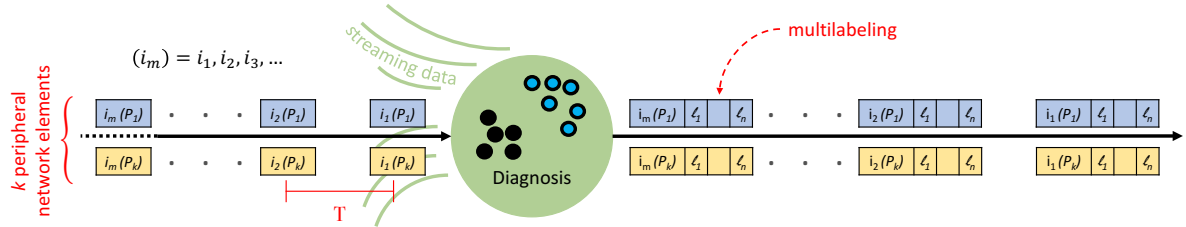


Figure 3.7 Diagnosis conditions.

Each instance will have C possible combinations of labels. This number is defined by equations (2) and (3), where B_i represents all the possible labels which represent the state of the layer i of the monitored network.

$$B_i = \text{class set of SOFI dataset labelled with failures caused in layer } i \quad (2)$$

$$C = \frac{(\sum_{i=1}^n \text{card}(B_i))!}{n! (\sum_{i=1}^n \text{card}(B_i) - n)!} - \sum_{i=1}^n \frac{\text{card}(B_i)!}{n! (\text{card}(B_i) - n)!} ; n = \text{number of layers} \quad (3)$$

For example, if we have two internal layers of the monitored network, there are two label sets as equation (4) defines. Suppose each one has two possible labels. Then, equation (5) produce the number of possible combinations taken in pairs (because there are two internal layers) between the two sets.

$$\begin{aligned} A &= \{a | a \text{ is a class of failure caused in layer } A\} = \{N, FA\} \\ B &= \{b | b \text{ is a class of failure caused in layer } B\} = \{N, FB\} \\ N &= \text{No failure}, \quad FA = \text{Failure in layer } A, \quad FB = \text{Failure in layer } B \\ \text{card}(A) &= 2, \quad \text{card}(B) = 2 \end{aligned} \quad (4)$$

$$\begin{aligned} C &= \frac{(\text{card}(A) + \text{card}(B))!}{2! (\text{card}(A) + \text{card}(B) - 2)!} - \left(\frac{\text{card}(A)!}{2! (\text{card}(A) - 2)!} + \frac{\text{card}(B)!}{2! (\text{card}(B) - 2)!} \right) \\ &= \frac{4!}{2!} - \left(\frac{1}{(0)!} + \frac{1}{(0)!} \right) = 6 - 2 = 4 \end{aligned} \quad (5)$$

On the other hand, the collecting of SOFI instances takes place every fixed time period ad infinitum; consequently, the Diagnosis module, as the container of the learning model, receives an infinite stream of SOFI instances (i_i) with a lapse of time in between them. So, the algorithm has to operate indefinitely, deal with large volumes of data, and needs to be tailored to the time available to perform failure classification for each instance ($t_{classification} < T$).

This condition is facing us to several challenges depicted in Figure 3.8. Using a limited amount of memory, processing instances faster than they arrive dealing with each one within a fixed interval and updating the model in such a way that it is not necessary to store new data inexorably. These aspects cover the typical data stream mining learning scenario, so it is challenging on its own.

Such difficulties are amplified in networking by the fact that the time elapsed in the normal state of a network is greater than in a fault state. So, the number of instances that we will classify as a failure will be significantly less than the instances that represent the normal network behavior (as evidence, the class ratio in the SOFI dataset is approximately 1:70). This phenomenon is known as class imbalance and is a problem generally associated with concept-drift (Z. Li et al., 2020).

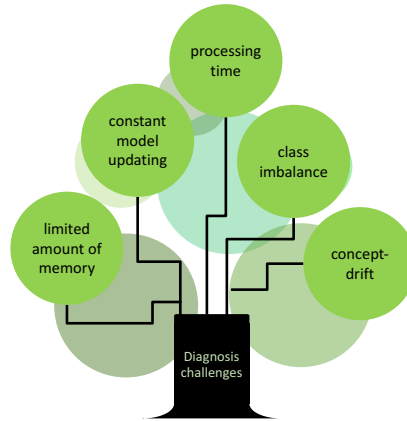


Figure 3.8 The ecosystem of challenges to be faced in diagnosis.

Incremental algorithms have emerged to deal with the frequent arrival of new data, which involves the three main aspects of the data stream learning scenario. Nevertheless, they alone are insufficient to support learning in imbalance and concept-drift contexts (as the experiment with SOFI dataset in section 3 of Chapter V demonstrates).

The last two issues merit further analysis to abord our specific problem. The next section explains the options for dealing with the challenges, and based on them, the components of the PALADIN diagnostic module, which meets the indicated initial conditions, will be presented.

4.1 How to tackle imbalance and concept-drift

There are three strategies to deal with concept-drift (Bifet et al., 2018). Firstly, as Figure 3.9(a) shows, using an external change detector algorithm (one or more) parallel with the main classifier over time. It measures stream properties (e.g., standard deviation, or instance distribution) to detect significant changes. When a change is detected, it triggers the revision and recalibration of

the current model. Secondly, as Figure 3.9(b) illustrates, feeding the model from estimators. An estimator monitors a statistic from the data stream, and the model synchronizes with these statistics. Thirdly, adopting an ensemble strategy, as Figure 3.9(c) depicts. It means implementing a management algorithm that selects a single or several model-building algorithms at different times. The management algorithm bears the responsibility of detecting and reacting to change and execute rules that enables it to create, erase, or revising the models. These three strategies can be implemented alone or a combination of them.

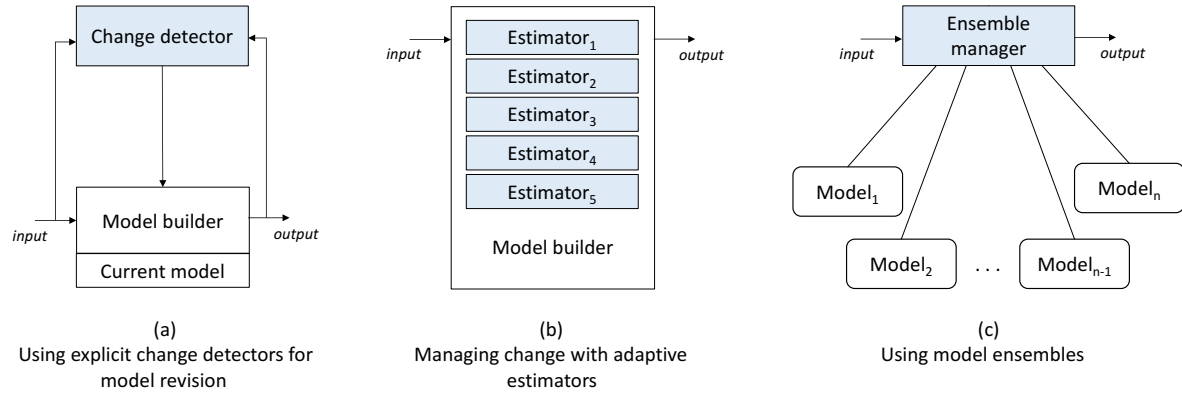


Figure 3.9 Strategies to deal with concept-drift. (Bifet et al., 2018)

On the other hand, our fault diagnosis scenario is also affected by class imbalance. The imbalance is a common and widely studied topic in traditional machine learning problems. It is addressed through well-defined techniques, such as over-sampling¹, under-sampling², a combination of both, and penalized models³. Nevertheless, deal with skewed static datasets is still a difficult task, so for skewed data streams, it gets even worse. Consequently, other unique characteristics of this last context must be taken into account.

Those characteristics are defined by (Fernández et al., 2018a): Firstly, not only concept-drift over time but also the ratio of skewed classes may change over time (in fact, our problem). Secondly, the class ratio may change over time, and even it could change so much that the imbalance ratio would reverse. Nevertheless, our research problem does not embrace this characteristic because it is well-known that failure instances will always be a minority class. Thirdly, new classes may emerge or existing ones to arrive less frequently, even disappearing. This aspect does not concern us for now as our objective is to determine whether or not there is an internal fault. The increase in diagnostic granularity is future work. Fourth, the overlap between classes can increase, most notably if concept-drift occurs, so a classification task is more complicated to perform. This last issue could potentially harm us.

As a result, concept-drift and imbalance in data stream blend several limitations that give rise to an emerging paradigm within machine learning.

¹ Over-sampling refers to add copies or generate synthetic samples from the minority class.

² Under-sampling refers to delete samples from the majority class.

³ Penalized models refer to punish the algorithm when it makes classification mistakes with minority class while training. So, the algorithm is being forced to increase its learning from the minority class.

A comprehensive study of several works that propose approaches to deal with this double challenge and all the difficulties that imbalance entails, is presented in (Fernández et al., 2018a). It is found that there is no a widely adopted algorithm to address the issue, so the authors who have faced it adapt existing techniques for static datasets to contexts where the data arrive sequentially to infinity.

Learning from imbalanced and drifting data streams is still relatively new. Many related issues still await to be appropriately analyzed, understood, categorized, and addressed (Fernández et al., 2018a). So, we decided to adopt the general framework proposed by (S. Wang et al., 2013), which has been the baseline for emerging works and fits our diagnosis problem. The next section explains its components in detail.

4.2 Diagnosis components

The paper (S. Wang et al., 2013) contends that imbalanced data streams require combined monitoring of class ratio and drift detection over time, and from this, proposes a general framework for learning from imbalanced data streams. Figure 3.10 shows this framework, which also represents the diagnosis module for the PALADIN model.

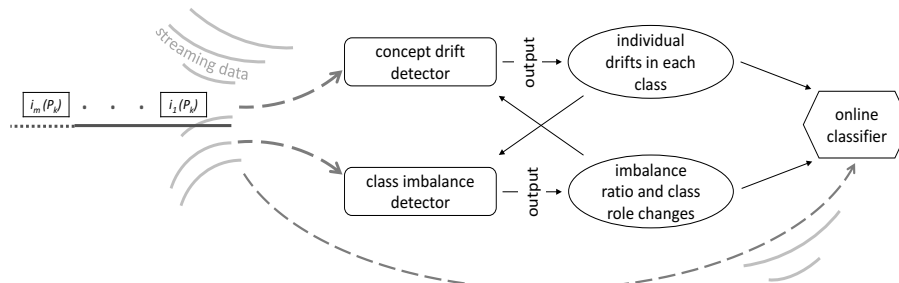


Figure 3.10 Diagnosis components (S. Wang et al., 2013).

There are three principal components, the concept-drift detector, the class imbalance detector, and the online classifier. These components communicate with one another for information on the current status of the data stream.

On the one hand, the class imbalance detector brings the class imbalance status. It detects which classes belong to the majority and minority, what is the current class ratio, and thus, which classes need more attention from the classification algorithm.

On the other hand, the concept-drift detector aims to detect the changes in the data distribution. It must be able to discover different types of drifts (sudden, gradual, incremental, among others). It focuses more on the minority class thanks to the class imbalance detector's output information, and it warns that imbalance detector about possible changes in the skew.

Finally, the online classifier decides when and how to respond to the imbalance and concept-drift based on the above components' information. The "how" refers to using a technique to cope with imbalance or drift, such as some resampling method. Of course, it is also responsible for making a real-time prediction.

The above framework (also our diagnosis components) addresses the ecosystem of challenges presented in the previous section because it incorporates two elements to directly tackle the problem of skewed data streams and concept-drift over time. On top of that, it defines an online learning model that, by its nature, can deal with a limited amount of memory, and constant model updating.

Section 3 of Chapter V details the results of the proposed components implementation using the SOFI dataset as the data stream. The experiment was carried out through twenty-five different incremental or online learning algorithms, the algorithm ADWIN (Bifet & Gavalda, 2007) as a concept-drift detector and SMOTE (Chawla et al., 2002) that is one of the most used and efficient rebalance methods. You can see in detail the tests, the analysis, and the result in the mentioned section, which reflect the good performance of the framework for the correct diagnosis of faults. The implementation of this module also represents the reaching of the second specific objective of the thesis.

Chapter IV: Dataset Construction

*"It is a capital mistake to theorize before one has data.
Insensibly one begins to twist facts to suit theories,
instead of theories to suit facts."
(Sherlock Holmes in A Scandal in Bohemia,
author: Arthur Conan Doyle, 1891)*

This chapter describes the process of building a dataset of symptom-fault causal relationships for an IP-based network, from now *SOFI* (**S**ymptom-**F**ault relationship for **I**P-Network) dataset. This process is based on *CRISP-DM* methodology (Chapman et al., 2000), which defines a six-phase reference model for data mining projects shown in Figure 4.1. Three of them refer to dataset construction, and they are our steps for building the dataset thanks to the fact that the sequence of the model phases is not rigid: business understanding, data understanding, and data preparation.

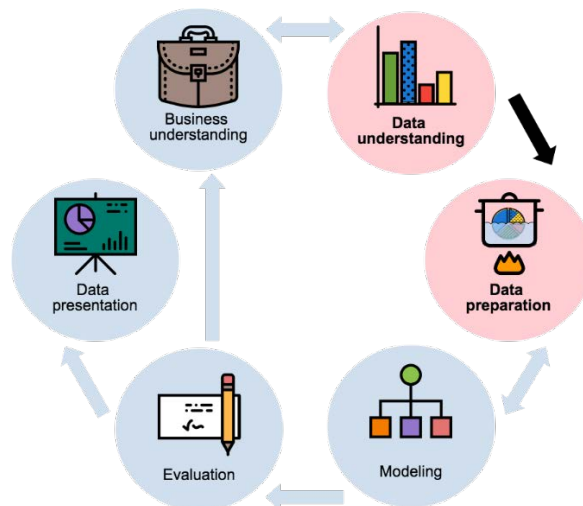


Figure 4.1. Phases of the CRISP-DM reference model.

The initial step is business understanding, which is focused on dataset objectives or data mining problem definition. In this respect, the objective of this work is to obtain data that allow relating faults of the internal network elements of the monitored IP-network with the symptoms observed in the peripheral network elements.

The second step is data understanding, which refers to the data collection and exploration processes. At this point, the collected information is raw data. Finally, the third step is data preparation, which covers the activities needed to construct the SOFI dataset from that raw data. These last two steps are described in greater depth in the following sections.

Another essential point to note is that the first specific objective of this work is met with the SOFI dataset construction.

1 Data understanding

The data understanding is the fundamental basis for the *SOFI* dataset construction because it represents the ground truth of failures behavior in an IP-network. This step is composed of three tasks, as shown in Figure 4.2. Firstly, in the data collection and description task, the monitoring data from an IP-network is acquired (traffic, logs, *SNMP* polling and traps) in order to have the dynamic view of the IP-network. Secondly, in the data exploring task, the examination of data characteristics, the attributes distribution analysis, and the relationship between attributes and internal events, are performed. Thirdly, the quality verification aims to detect unsent attributes during monitorization or with an empty value. These tasks are described in detail in the next subsections.



Figure 4.2. Data understanding activities.

1.1 Fault data collection and description

The monitoring data collection was performed in an extensive campus network through different network monitoring systems which supports the traditional and more used management standards, such as *SNMP*, Syslog, and sFlow or NetFlow. Several link failures with impact (the traffic significantly goes down during failure) were induced at different instants, in order to collect healthy and in-failure network behavioral patterns. Data are extracted directly from the monitoring systems storage used.

Within this chapter, section 1.1.1 describes the used network scenario. Section 1.1.2 exposes the fault data collection experiment. Section 1.1.3 presents the process for extracting raw monitoring data and describes all the assembled data.

1.1.1 Network scenario

A complete large enterprise network (refer to section 1 of Chapter I) was emulated in order to have an accurate knowledge about failures and total control over them. The campus infrastructure is the

monitored network on which fault diagnosis is made, so the peripheral network elements are the core layer devices (see Figure 4.3).

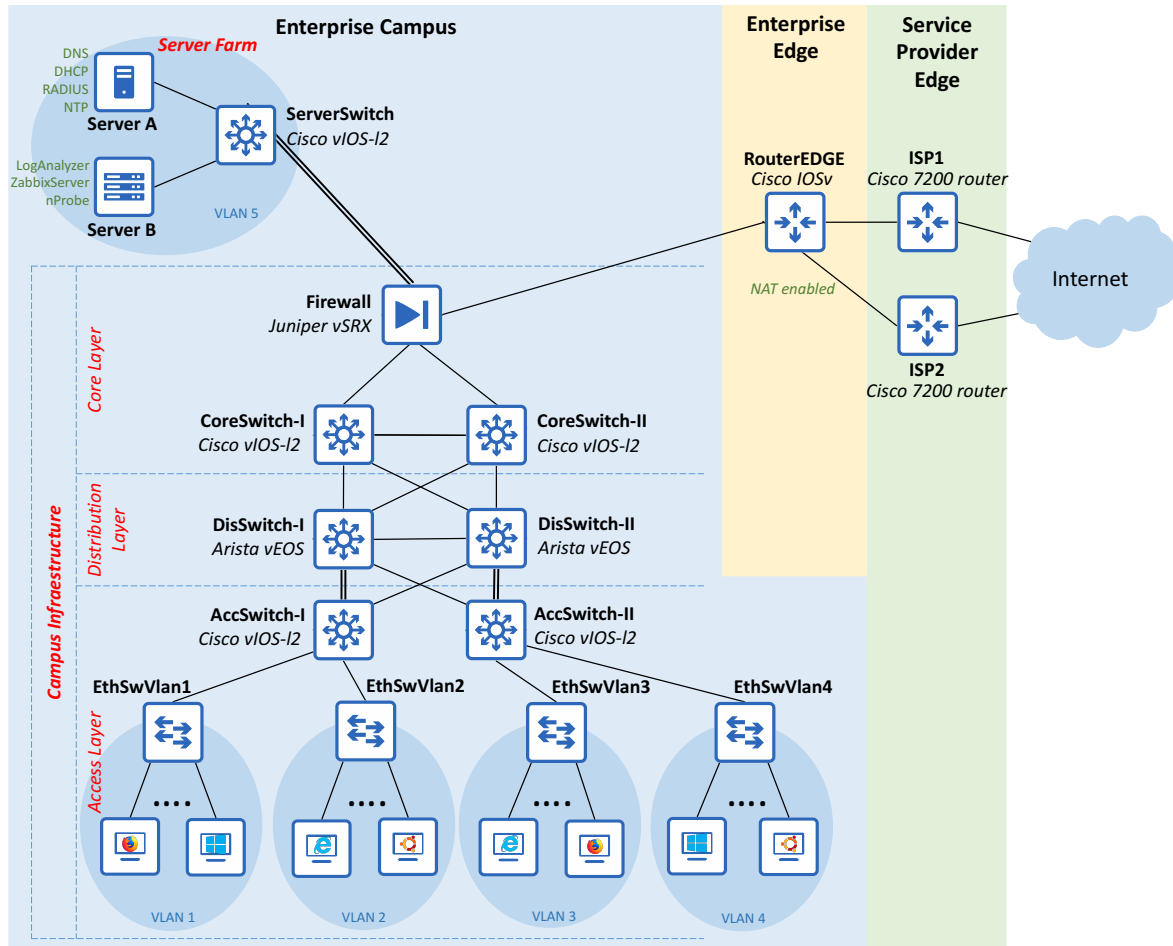


Figure 4.3 Topology of the Emulated Enterprise Network.

The emulated network allows collecting real monitoring data because of it provides Internet connection, and all the carried traffic is generated from clients with access capabilities to the external network (such as *DHCP* and *DNS* requests, browser, etc.).

As shown in Figure 4.3, the implemented campus module emulates all the campus infrastructure layers enabling access to about more than one thousand clients through four *VLAN*. All campus infrastructure devices are able to respond to *SNMP* requests, send event messages to a logging server by Syslog protocol, and exports flow records towards a server collector by Netflow/sFlow protocol.

In this network, the server farm only deploys the network services and the management block, so a simple data center was emulated collapsing the network layers into a single switch. Server A provides all the network services, and Server B contains the monitoring software tools.

Because of the emulated network only provides clients with Internet access, an edge distribution is not necessary; moreover this block is optional as shown in section 1 of Chapter I.

For the same above reason, the implemented enterprise edge module only emulates the Internet functional area by an edge router. Finally, the implemented service provider edge module emulates two different *ISP*. Both *ISP* routers are bridged to the Ethernet Card of the machine in which emulation runs, in order to get a real connection to the Internet.

As the emulation tool, the Graphical Network Simulator 3 (*GNS3*) was selected because it is an open source, free real-time network software emulator, used by hundreds of thousands of networks engineers and companies all over the world (such as AT&T, Google, NASA, and Exxon). It allows the emulation, configuration, testing, and troubleshooting of virtual and real networks. *GNS3* virtualizes real hardware devices from multiple network vendors, whether commercial or open source (such as Cisco, Juniper, HP, and Fortinet), so enables engineers to test interoperability between many vendors and to combine several network technologies in complex networks (such as *SDN*, *NFV*, and Linux) (Bombal & Duponchelle, 2019).

The components of each module of the enterprise network and the emulation hardware and software specifications are described below. Annex A contains all the detailed configuration (subnetting, servers, network address translation, etc.) for each device.

1.1.1.1 Enterprise campus module

The enterprise campus has four *VLAN* in which several client types are emulated through different types of virtual machines or container; client types are: Basic PC, Ostinato (packet generator), Firefox, Windows 10, PC Loopback and Ubuntu. Each client uses *DHCP* service to get an *IP*, except those of *VLAN4* which have a static network configuration for management reasons. Each Ethernet switch represents a *VLAN*; for example, clients connected to EthSwVlan1 are part of *VLAN1* and so on. Thanks to segmentation to *VLAN*, client's traffic is sent by a corresponding access switch to the distribution layer without being spread across the other access switches on the campus.

For high availability, the access switches have a redundant uplink connection. This dual attached ensures the access switches remain connected to the distribution layer, even in case of failure of one of the distribution switches.

There are two emulated distribution switches which connect the lower layer to the core and route the traffic between clients. The redundancy in this layer allows having two equal-cost paths to each destination network. According to the OSI model, these devices work in both layer two and three. Their interfaces connecting the access layer are switch port or layer two interfaces while their interfaces that connect the core are layer three interfaces which use the OSPF protocol to route traffic.

The core of the campus infrastructure is composed of two core switches and a firewall. The firewall filters and inspect traffic for forwarding it to the outside or inside campus infrastructure according to administrative rules (for example, request DNS are allowed only from inside infrastructure campus to data center/server farm). It also has the responsibility of connecting the enterprise campus module with the enterprise edge module.

The emulated server farm contains a multilayer switch (ServerSwitch) and two servers which make up the *VLAN5*. Server A house the network services: *NTP*, *DHCP*, *DNS*, and *RADIUS*. Server B deploy the monitoring software tools: Zabbix for *SNMP* probing, Rsyslog for collect Syslog messages,

LogAnalyzer to display from a client *GUI* the collected logs, and nProbe for collect traffic flow (any monitoring tool can be used. It is not intended to have the best tools but those commonly used by network administrators. The objective is to emulate a real-world business network. For this particular emulation, free tools were selected). The firewall grants permission to clients of *VLAN4* for access to monitoring tools and managing the network.

All the campus network elements synchronize their time with *NTP* server, allow the management by *RADIUS* protocol, deal with *SNMP* requests, and send Syslog message and traffic flows to the corresponding servers.

Table 4.1 presents the specification for each network element of the enterprise campus module.

Table 4.1 Network elements of enterprise campus module.

Submodule or Block	Network Element		Vendor	Software Specifications
Building Access	Access Switches		Cisco	Cisco vIOS-l2 Qemu appliance on qcow2 disks (version 15.2)
	Ethernet Switches		GNS3	Ethernet swithes are part of the Dynagen package used by GNS3.
	Clients	Basic PC	Linux	Core Linux Qemu Appliance (version 4.7.7)
		Ostinato	Linux	Ostinato Qemu Appliance (version 0.9.1)
		Firefox	Linux	Firefox Qemu Appliance (version 31.1.1)
		Windows 10	Windows	Windows10 VMWare Appliance (version w/Edge)
		PC Loopback	Linux	Loopback connection
		Lubuntu	Linux	Lubuntu 17.10 installed in VMWare
Building Distribution	Distribution Switches		Arista	Arista vEOS Qemu appliance on VMware disks (version 4.17.8M)
Campus Core	Core Switches		Cisco	Cisco vIOS-l2 Qemu appliance on qcow2 disks (version 15.2)
	Firewall		Juniper	Juniper vSRX Qemu appliance on qcow2 disk (version 17.3R1)
Server Farm	Server Switch		Cisco	Cisco vIOS-l2 Qemu appliance on qcow2 disks (version 15.2)
	Server A		Linux	O.S.: Ubuntu 16.04.4 LTS Servers: ISC DHCP server Bind server (for DNS service) NTP server FreeRADIUS
	Server B		Linux	O.S.: Ubuntu 16.04.4 LTS Servers: nProbe (version 8.4.180628) Zabbix server appliance (version 3.4.2) Rsyslog LogAnalyzer (version 4.1.6)

The sections A.1, A.2, A.3 and A.4 of Annex A describe the internal configurations of all the devices in the Enterprise Campus Module.

1.1.1.2 Enterprise edge module

An edge router was emulated as a boundary between the enterprise campus module and the service provider module. Also, it enables the campus to connect to the Internet.

In a real network and certainly in this emulated network, the service providers assign a public address range to the enterprise, but a wider private address range is used inside the campus. The edge router has enabled the Network Address Translation (*NAT*), to deal with these incompatible addresses. Finally, the Border Gateway Protocol (*BGP*) is used to exchange routing information with both *ISP*.

Table 4.2 presents the specification for the network element of the enterprise edge module.

Table 4.2 Network elements of enterprise edge module.

Network Element	Vendor	Software Specifications
EDGE router	Cisco	Cisco IOSv Qemu appliance (version 15.6(1)T)

The section A.5.1 of Annex A describes the internal configurations of the device in the Enterprise Edge Module.

1.1.1.3 Service provider edge module

Two high-performance routers were emulated to play the Internet provider role. These routers are bridged to the interface corresponding to the *NIC* of the machine over which the emulation runs.

Table 4.3 presents the specification for the network element of the service provider edge module.

Table 4.3 Network elements of service provider edge module.

Network Element	Vendor	Software Specifications
ISP1	Cisco	Cisco 7200 router (IOS version 15.2(4)M7)
ISP2		

The section A.5.2 of Annex A describes the internal configurations of the devices in the Service Provider Edge Module.

1.1.1.4 Hardware and Software specifications

All devices are emulated and distributed in two *GNS3* servers. Each server runs over an independent machine, and there is one *GNS3* user interface through which the devices are interconnected and configured. After a preliminary implementation in the network laboratory at UC3M, the *GNS3* servers were installed on virtual machines running over the Telco 2.0 laboratory of the Telematics Department of Universidad del Cauca which is composed of several blade servers with 100 cores, 768 GB RAM, and 24 TB hard disk.

Figure 4.4 shows the details of the machines. The hardware and software specifications for each machine are listed below.

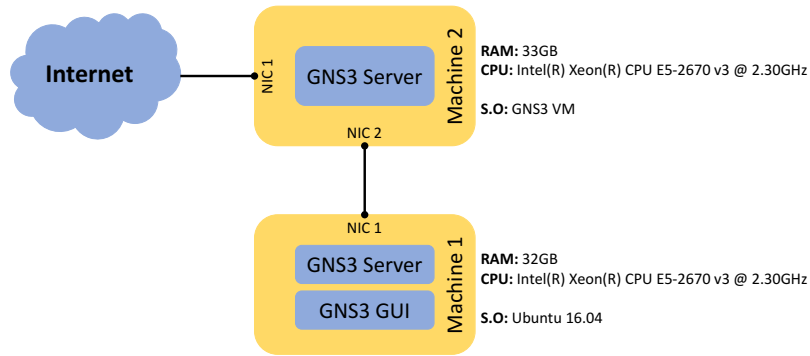


Figure 4.4 Emulation Requirements.

Hardware Specifications:

Machine 1:

RAM: 32 GB
 Processor model: Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30 GHz
 CPUs: 6

Machine 2:

RAM: 33 GB
 Processor model: Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30 GHz
 CPUs: 8

Software Specifications:

Machine 1:

Ubuntu 16.04, GNS3 version 2.1.3, Vmware and VIX API, and Docker version 17.12.0-ce

Machine 2:

GNS3 VM version 0.10.14 (with GNS3 version 2.1.6, KVM support available and cloud node support)

Table 4.4 presents the memory allocation for all emulated devices.

Table 4.4 Memory allocation for emulated devices.

Module		Network Element	Assigned RAM
Enterprise Campus	Access	Basic PC	256 MB
		Ostinato	256 MB
		Firefox	512 MB
		Windows 10	1024 MB
		Lubuntu	1024 MB
		AccSwitch-I	2048 MB
		AccSwitch-II	2048 MB
	Distribution	DisSwitch-I	3072 MB
		DisSwitch-II	3072 MB
	Core	CoreSwitch-I	2048 MB
		CoreSwitch-II	2048 MB
		Firewall	8192 MB
	Server Farm	Server A	2048 MB
		Server B	2048 MB
		ServerSwitch	1024 MB

Enterprise Edge	RouterEDGE	3072 MB
Service Provider Edge	ISP1	2048 MB
	ISP2	2048 MB

1.1.2 Data collection experiment

The data collection was performed through the following three consecutive steps.

- **Monitoring tools configuration step:** For this purpose, with the in-service emulated network, the Zabbix server was configured with official Zabbix templates for collecting all the generic SNMP parameters.

According to the template, each parameter type is periodically collected with a different period, also called the polling interval, and they have associate one or more triggers. For status descriptors, the period oscillates between 30 and 60 seconds (such us, device uptime, and availability). For traffic descriptors, the period oscillates between 180 to 300 seconds (such us, bits sent, inbound packets with errors, and outbound packets discarded). For identification parameters, the period is set to 3600 seconds (such us, device name, and device description).

The above default time conditions limit the construction of complete instances for each measuring. So, it was necessary to set the same period for collecting all the parameters. This polling interval is 180 seconds or 3 minutes. The time decision was based on selecting a midpoint between the possible times, and on estimating a possibly time chosen by a real-world network manager.

The tools Rsyslog and nProbe were executed with defect configurations because these are listening servers. The first one stores the arriving logs in a simple database. The second one uses comma-separated formatting to save traffic flows.

It is essential to mention that although the *SOFI* dataset consists of peripheral network element parameters, the collection is also done for internal elements. Undoubtedly it is necessary to have the ground truth to corroborate the fault propagation as well as to cross any essential information to verify the dataset.

- **Selection of link failures step:** Based on different works like (Gill et al., 2011) and (Potharaju & Jain, 2013), different faults associated with link failures were listed to analyze how to induce them into the emulation. The Table 4.5 summarizes this analysis.

In order to eliminate many spurious failures from our analysis and focus on problems that have a measurable impact on the network, those listed faults were induced for different timescales (5, 12, and 17 minutes), and the network behavior was observed while clients navigated around several Web pages or Internet services. From this observation, the faults "link flapping" and "unexpected reloads" did not cause an impact failure because from the user point of view the network never crash, and they are absent from the actionable

network logs (seen in LogAnalyzer and Zabbix events). Consequently, these two faults will not be induced during data collection.

In parallel, we can conclude that our emulated network provides a good approximation of the real world, because of those two faults are not observable from monitoring tools and do not impact connectivity in a real network either (Gill et al., 2011).

Table 4.5 Faults to induce.

No.	FAULT	HOW TO INDUCE FAULT	FAILURES OCCUR
1	CRC packet error	N.P	
2	Faulty cables	Removing a link connection.	✓
3	Fiber cuts	Removing a link connection.	✓
4	High link utilization	Injecting a lot of traffic from a client.	✓
5	Line card failure	Shutting down interfaces from line card.	✓
6	Link down	Shutting down one or both linked interfaces.	✓
7	Link flapping	Shutting down and starting up one or both linked interfaces, several times.	✗
8	Misconfigurations	Removing a configuration line or invert a configuration line.	✓
9	Operative system bugs	N.P	
10	Protocol issues	A cable disconnected or incorrectly connected can produce incorrect routes.	✓
11	Soft-parity errors	N.P	
12	Software bugs	N.P	
13	Unexpected reloads	Rebooting virtual machine of the network element.	✗
14	Distribution layer going down	Shutting down the access switches linked of the same distribution switch for a while, or shutting down the interfaces, of these access switches, connected to the distribution layer for a while.	✓
N.P: it is not possible to induce it.			

- **Link failures induction step:** For this purpose, different network traffic behaviors are first generated, each one for an extended period of time: a) clients surfing the web, b) clients consuming social networks and surfing the web, c) clients watching online video, d) all the above. The network behaviors above are an attempt to change the network utilization rate and to allow a comprehensive analysis before, during, an after failure; so, network behavior is occasionally changed. Then, within each behavior, different failures are induced randomly chosen with several durations. Between each induction, there is at least the necessary time to failure recovery. All the induced failures are documented in order to have a well-known failure set. So, we register the type, the device that causes the failure, and the time interval for each induced failure.

1.1.3 Data extraction and description

As shown in Figure 4.5, the network has both a static and dynamic view. We define the static view as the network configuration, and the dynamic view as the performance in time, so it is collected periodically.

The dynamic view is the base for *SOFI* dataset construction and is composed of four data types. Firstly, the network event logs that are extracted from log messages, *SNMP* polling, and traps. Secondly, the task coordination documents which are annotations or records about incidents (trouble tickets). They amount to the experiment's well-known link failures. Thirdly, the network traffic data which is the amount of data transferred on network interfaces. It is extracted from *SNMP* polling and flows captured via NetFlow/sFlow protocol. Finally, the maintenance data report track activities such as upgrades, configuration changes, and repairs.

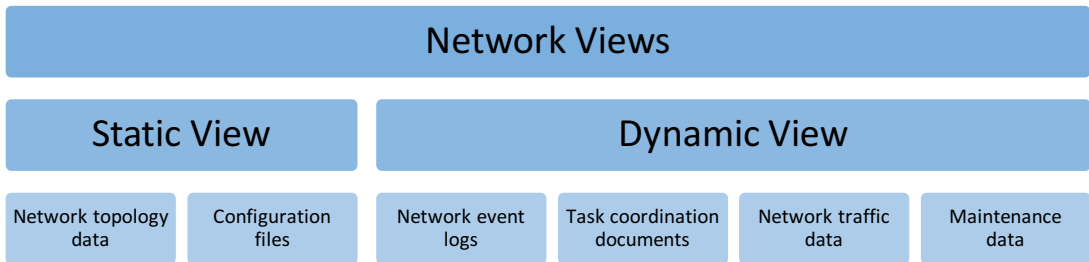


Figure 4.5 Network views.

Consequently, two data types must be collected in the emulated network: the network event logs, and the network traffic data. Figure 4.6 shows the elements involved in these data collection.

As shown, the traffic flows are stored in comma-separated text files with the extension *.flow*, and with a well-defined structure, so it is not necessary to perform extra tasks for their extraction. Each flow is described by 116 parameters according to Netflow v9 standard (Claise, 2004). Table B.1 of Annex B presents the descriptors list.

The network logs are stored in a simple database, and their extraction does not require complex queries (it is a single table). Each incoming system log from the devices is described by the parameters list of Table B.2 of Annex B and are based on Syslog protocol messages standardized by *RFC 5424* (Gerhards, 2009).

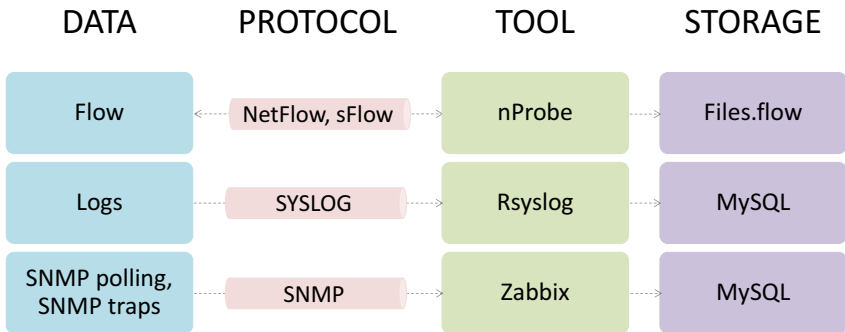


Figure 4.6 Elements involved in network event logs collection and network traffic data collection.

On the contrary, the data related to *SNMP* requests are stored in a large complex database, so it was necessary to design a software tool to extract them. The collected *SNMP* data is described in table B.3 of Annex B sorted by three types: traffic descriptors, status descriptors, and identification descriptors. The first type contains parameters related to traffic and are collected from all the interfaces of each device. The second and third data types are related to the whole device.

Annex C describes the said extraction tool, and Figure 4.7 shows the structure of the extracted data. The extracted data conforms a comma-separated text file, where its attributes are the parameters collected, and the timestamp reflects when they were collected. This means that for each timestamp registered in the parameter's history, there is an instance. So, each parameter is recovered with its timestamp. If more than one parameter has the same timestamp, they set in the same instance. If a parameter is not recovered for a timestamp, its value is set to -1. This is done separately for each device of the monitored network.

timestamp,	parameter ₁ ,	parameter ₂ ,	parameter ₃ ,	...	parameter _n
t ₁ ,	-1,	v ₁₂ ,	-1,	...	v _{1n}
t ₂ ,	v ₂₁ ,	-1,	v ₂₃ ,	...	-1
.					.
.					.
.					.
t _n ,	v _{n1} ,	-1,	v _{n3} ,	...	-1

Figure 4.7 Structure of the *SNMP* extracted data.

The structure of the *SNMP* extracted data is the base for *SOFI* dataset, so the set of files resulting from this extraction is called *SOFI* raw data. Flows and logs are used to explore the failure behavior, when a fault occurs, or to cross information for labeling. All data collected gathers the *SOFI* ground truth.

1.2 Raw data exploring

This section describes the two raw data exploration findings, the first related to the parameter delays in polling responses, and the second refers to the late triggering of internal events after failure.

1.2.1 Parameter delays

As shown in Figure 4.8(a), in data extraction, getting all the values of monitored parameters periodically was expected. In other words, a value set should have been registered at the same timestamp. Figure 4.8(b) shows the raw dataset structure expected, where each instance has all the parameter values, and the time between instances is three minutes.

Nevertheless, according to the observations, each measure is divided into several value subsets that arrive within a short time window. Meaning, there are delays between the same group of parameters even though they are polled simultaneously.

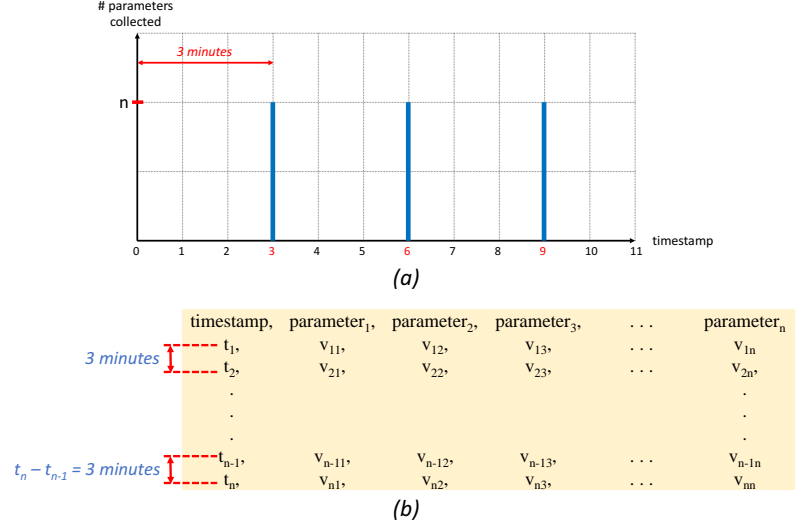
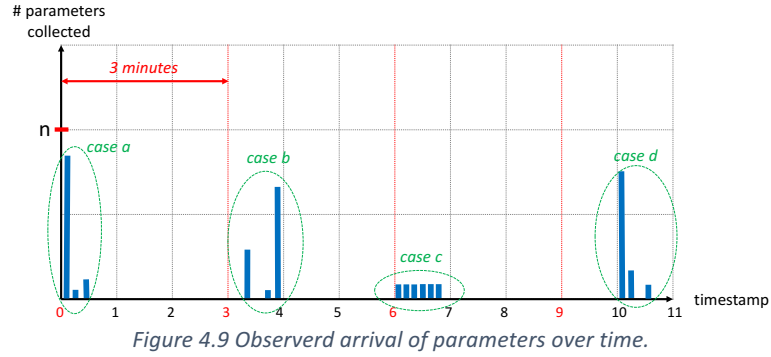


Figure 4.8 Expected periodic arrival of parameters.

As shown in Figure 4.9, there are several parameter distribution cases in time. The most common (“case a” in Figure 4.9), concentrates most of the parameters very close to the polling time and the rest of the parameters during the next seconds. For example, the polling time is 00:03:00, and the timestamp of the first measured value subset is 00:03:03. Exceptional and less frequent cases were observed (like case b, c, and d in Figure 4.9). For these cases, the concentration of most parameters is almost a minute after the polling time.



According to this behavior, the original raw dataset structure is as shown in Figure 4.10. Each polled parameter set is recorded in two or more instances, so the time between instances is not fixed. If two instances belong to the same parameter set, the time between them is much less than three minutes. The time between the first instances of two consecutive parameter sets is approximately three minutes ($t_4 - t_1$ in Figure 4.10).

	timestamp	parameter ₁	parameter ₂	parameter ₃	...	parameter _n	
$t_4 - t_1 \cong 3 \text{ minutes}$	t_1	v_{11}	-1	-1	...	v_{1n}	group a
	t_2	-1	v_{22}	-1	...	-1	
	t_3	-1	-1	v_{33}	...	-1	
	t_4	v_{41}	v_{42}	-1	...	-1	group b
	t_5	-1	-1	v_{53}	...	v_{5n}	
	
	
	
$t_n - t_{n-1} \neq 3 \text{ minutes}$	t_{n-1}	-1	-1	v_{n-13}	...	-1	parameter values matrix
	t_n	v_{n1}	v_{n2}	-1	...	v_{nn}	
$t_n - t_{n-1} < 3 \text{ minutes}$							

Figure 4.10 SOFI raw dataset structure.

In other matters, let's follow the example of values distribution shown in the matrix of Figure 4.10. We extract the matrix of values of parameters. This matrix is named V . Then, let's suppose that the first three rows compose a polled parameter set, as Figure 4.11(a) indicates. If these rows are extracted to a new matrix named A , each column of it has at most one value that must be greater than or equal to zero for numeric parameters, or one value that must be different from null for text parameters. We can express A as n columns-vectors (see Figure 4.11(b)), where n is the number of parameters collected. Each column-vector represent an element set, as shown in Figure 4.11(c). From observations, a vector whose elements are the maximum values from these sets represents the parameter set collected in one polling (see Figure 4.11(d)).

$$\begin{aligned}
 (a) \quad V &= \begin{pmatrix} v_{11} & -1 & -1 & \dots & v_{1n} \\ -1 & -1 & v_{23} & \dots & -1 \\ -1 & v_{32} & -1 & \dots & -1 \\ v_{41} & v_{42} & -1 & \dots & -1 \\ -1 & -1 & v_{53} & \dots & v_{5n} \\ \dots & \dots & \dots & \dots & \dots \\ v_{m1} & -1 & v_{m3} & \dots & v_{mn} \end{pmatrix} \\
 (b) \quad A &= \begin{pmatrix} v_{11} & -1 & -1 & \dots & v_{1n} \\ -1 & -1 & v_{23} & \dots & -1 \\ -1 & v_{32} & -1 & \dots & -1 \end{pmatrix} \\
 A &= (\vec{a}_1 \ \vec{a}_2 \ \vec{a}_3 \ \dots \ \vec{a}_n) \\
 (c) \quad A_1 &= \{a | a \text{ is an element of } \vec{a}_1\} \\
 A_2 &= \{a | a \text{ is an element of } \vec{a}_2\} \\
 A_3 &= \{a | a \text{ is an element of } \vec{a}_3\} \\
 &\vdots \\
 A_n &= \{a | a \text{ is an element of } \vec{a}_n\} \\
 (d) \quad \vec{g} &= (\max(A_1) \ \max(A_2) \ \max(A_3) \ \dots \ \max(A_n)) \\
 \vec{g} &= (v_{11} \ v_{32} \ v_{23} \ \dots \ v_{1n})
 \end{aligned}$$

Figure 4.11 Analysis of the obtained matrix of parameter values.

So, to find the instances of the same group of parameters in a SOFI raw dataset, we extract the V matrix (equation (6)) of $m \times n$ dimensions, where m is the number of raw instances and n the number of monitored parameters. Given a matrix V , we can get as many A matrices as possible where each one conforms to equation (7). In equation (7), b and k terms are the row numbers of the first and last instance, which make up the group of instances that represent a set of parameters from one polling. T is the polling period (3 minutes in this particular case). From each A matrix, we get a vector \vec{g} by equation (8), where \vec{g} is a complete instance of parameters from one polling.

$$V = [v_{ij}]_{1 \leq j \leq n}^{1 \leq i \leq m} \quad (6)$$

$$A = [v_{ij}]_{1 \leq j \leq n}^{b \leq i \leq k}, \text{ where } (t_k - t_b < T) \wedge (t_{k+1} - t_b \geq T) \quad (7)$$

$$\vec{g} = [\max(A_j)]_{1 \leq j \leq n}, \text{ where } A_j = [v_{ij}]_{b \leq i \leq k} \quad (8)$$

The dataset labeling is a process based on both the instances timestamp and well-know failures timestamp. So, it is necessary to assign a suitable timestamp for each \vec{g} vector found. This issue is further explained in section 2.1.1 Time-based grouping.

Furthermore, we noted that the dimension of *SOFI* raw dataset depends on the number of peripheral element interfaces. Therefore, the features set varies depending on the peripheral level selected. The solution to this issue is explained in section 2.1.2 Interface-based grouping.

1.2.2 Late triggering of internal events

It is expected that in the polling immediately after the moment of a fault induction, parameter values collected from the internal devices involved in a failure induction could trigger an event in the monitoring system (Zabbix for this case). However, the exploration of raw data indicated a sequence of events triggered at unexpected times related to those internal devices⁴.

Data exploration found that one or more events are triggered in the monitoring system at the induction level after inducing the failure. Figure 4.12 represents the sequence observed. The figure shows two timelines. The upper one indicates the moment of fault induction (F), the moment of fault recovery (F'), and the events ($e_1 \dots e_n$) triggered by parameters values from the device that causes the failure (internal element)⁵. A green downward arrow represents the beginning of an event, while a green upward arrow represents its end. Green dotted lines indicate the duration of each of those events. Because of the granularity per minute applied in this work, the figure indicates the minutes of induction and recovery of faults, shaded with blue, and projected to the second timeline. The total duration of the fault is shaded with light blue. The lower timeline shows the moments when we observe sudden changes of parameter values in the peripheral device, indicated by red asterisks. Finally, the red dotted lines are intended to indicate the delay between the occurrence of the fault and, at the upper timeline, the first event generated in the element that causes the failure and, at the lower timeline, the change of parameters monitored in the peripheral device.

As indicated by the delays between the induced failure and the reported events at the internal element, the failure is not detected in the expected timestamp, and there are even cases where events are never triggered. This issue is not harmful because the detection generally occurs; nevertheless, it affects the failure detection time so that it can involve risk for the monitored network.

⁴ It is assumed that behavior could be due to a mechanism to make sure something has indeed happened (the Zabbix documentation does not explain this behavior).

⁵ The triggered events are inferred from the behavior of the parameters. Some examples of those reported by the monitoring system (Zabbix) are "no SNMP data collection", "high ICMP ping loss", "high error rate [interfaces list] [vlan list]", and so on.

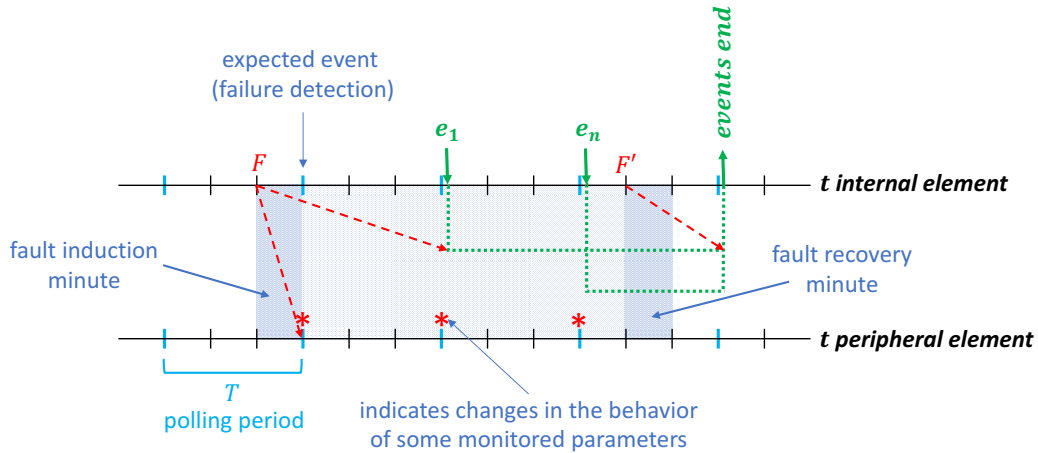


Figure 4.12 Behavior of induced fault.

On the other hand, as shown in Figure 4.12, although no events are generated at the peripheral layer during the failure, a change in the behavior of some monitored parameters is noticeable. Those changes take place in the polling immediately after the fault has been induced in the internal element, so we consider them as failures symptoms. Section 1 of Chapter V presents an in-depth analysis of parameter behavior during a failure. As a consequence, the peripheral symptom monitorization can significantly mitigate the observed late triggering of internal events and provide timely fault detection.

1.3 Data quality verification

We manually inspect the different dimensions of data quality (Loshin, 2011) in the *SOFI* raw data, focusing on those applicable to our context. This quality examination revealed three particular issues:

1. In almost every instance, two text type values were blank (*device_location* and *device_contact_details*), but it does not mean that they are missing values.
2. Three text type values almost always had the same values (*device_description*, *device_name*, and *system_object_id*) because they are the device's static descriptors.
3. Many parameters were set to -1; nevertheless, they do not represent missing values for the instance either; as described in section 1.2.1 of this chapter, this is since a parameter set polled at the same time can arrive dispersed in time.

Consequently, *SOFI* raw dataset does not present any significant quality problem, and the previous issues will be resolved in the data preparation process (section 2 of this chapter).

2 Data preparation: structuring

As shown in Figure 4.13, the *SOFI* dataset structuring process from raw data consists of three steps. The first step is to group those instances resulting from the same monitoring request, on account of delays between parameters of the same polling observed in the above data exploration. Section 2.1.1 presents a detailed description of this step.

The second step is to group parameters so that the *SOFI* dataset has a fixed dimension of attributes, regardless of the number of device interfaces. A dataset with a fixed dimension is essential for the learning model of the diagnostic process to be valid for any number of peripheral device interfaces, and therefore for any chosen monitored network. This process is further explained in section 2.1.2. *SOFI* dataset is the result of executing the above two steps, and the third step allows labeling it from a well-known failure set. Section 2.1.3 describes the labeling procedure.

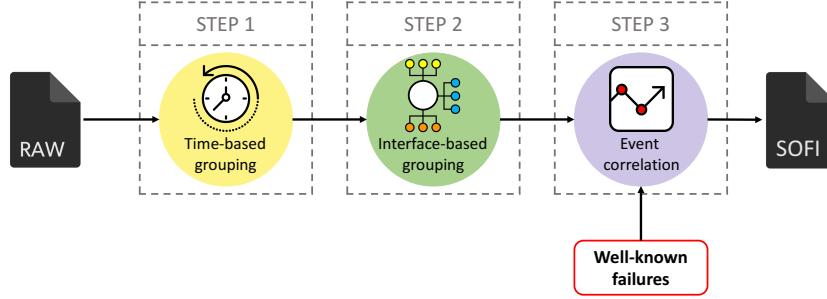


Figure 4.13 Steps for structuring *SOFI* dataset.

Each step represents a function that transforms a matrix into another one. Figure 4.14 shows the three functions. The first function $f(R)$ transforms a matrix R of size $u \times v$ into a matrix D of size $m \times (v + 1)$, where $m < u$ because of rows' grouping and a new attribute represents a distribution range. The second function $g(D)$ transforms the matrix D into a matrix S of size $m \times n$ where $n < v$ since the operations between columns make a dimensionality reduction. The matrix S is the *SOFI* dataset without the label. Finally, the third function $h(S)$ transforms the matrix S into a matrix L of size $m \times (n + 1)$, adding a class column. The matrix L corresponds to the labeled *SOFI* dataset.

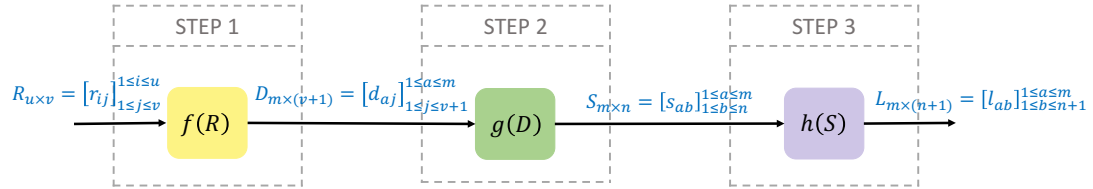


Figure 4.14 Functions for structuring *SOFI* dataset.

2.1.1 Time-based grouping

The raw data exploration showed different arrival times for a parameter set from the same polling because of their values are sent through different responses. Consequently, it is necessary to order those responses in the same instance of the dataset.

For this purpose, we consider the raw dataset as a matrix R of size $u \times v$, as defined in equation (9). This matrix can be partitioned into smaller matrices, so R can be expressed as m submatrices P_m , as shown in equation (10), where each submatrix obeys equation (11). A submatrix P_m represents a set of raw instances containing the same polling parameters, so the timestamps delimiting the matrix obeys the condition indicated in equation (11), where T is the polling period (3 minutes in this particular case).

$$R_{u \times v} = [r_{ij}]_{1 \leq j \leq v}^{1 \leq i \leq u} \quad (9)$$

$$R_{u \times v} = \begin{pmatrix} P_1 \\ \dots \\ P_m \end{pmatrix}; m < u \quad (10)$$

$$P_m = [r_{ij}]_{1 \leq j \leq v}^{b \leq i \leq k} \quad ; \text{ where } (r_{k1} - r_{b1} < T) \wedge (r_{(k+1)1} - r_{b1} \geq T), \\ \text{and } r_{i1} = \text{first parameter from } r_i \text{ instance (timestamp)} \quad (11)$$

On the other hand, the matrix D of size $m \times (v + 1)$ (see equation (12)) represents the dataset that stores the parameter set values from each polling in a single instance.

$$D_{m \times (v+1)} = [d_{aj}]_{1 \leq j \leq (v+1)}^{1 \leq a \leq m} \quad (12)$$

Equation (13) represents the partition of D into its row matrices d_m . Each row matrix d_m can be partitioned into three matrices, as shown in equation (14). The first matrix, Q , is a one-dimensional matrix with a unique element. This element is the timestamp that best represents the arrival time of all the one polling response messages. The second matrix M is a row matrix that contains all the arriving polling parameter values. The third matrix, Y , is a one-dimensional matrix with a new derived parameter.

$$D_{m \times (v+1)} = \begin{pmatrix} d_1 \\ \dots \\ d_m \end{pmatrix} \quad (13)$$

$$d_m = (Q \ M \ Y) \quad (14)$$

Each column of P_m (from column 2 to the last column) has at most one value that must be greater than or equal to zero for numeric parameters or one value that must be different from null for text parameters. From observations, the row vector whose elements are the maximum values of each column of P_m represents the parameter set collected in one polling. The matrix M comprises those maximum values of each column of P_m , as defined by equation (15).

$$M_{1 \times (v-1)} = [\max(C_j)]_{j \leq 2 \leq v} \quad ; C_j = \{c: \text{element of } P_m^{(j)}\} \quad (15)$$

On the other hand, the dataset labeling described in section 2.1.3 is a process based on both the instances timestamp and well-known failures timestamp. So, it is necessary to assign a suitable timestamp for every instance, so Q is essential. The matrix Q comprises the third quartile timestamp as the unique element (see equation (16)). The third quartile is derived from the values of the first column of P_m .

$$Q_{1 \times 1} = \text{the third quartil timestamp} \quad (16)$$

The choice of the third quartile timestamp follows the parameters' distribution over time and indicates when most parameters have arrived (75% of them). Two examples are given below in order to illustrate and understand the timestamp choice.

Suppose a poll to a peripheral device for 31 parameters, the polling time is t_0 and the device split its response into three messages which arrive in three different timestamps. Figure 4.15(a,b) illustrates the number of parameters collected for each timestamp. If the items are ordered by timestamp, as shown in Figure 4.15(c), it is possible to find the third quartile ($Q3$) of that set. The third quartile indicates the moment at which 75% of the data was already collected.

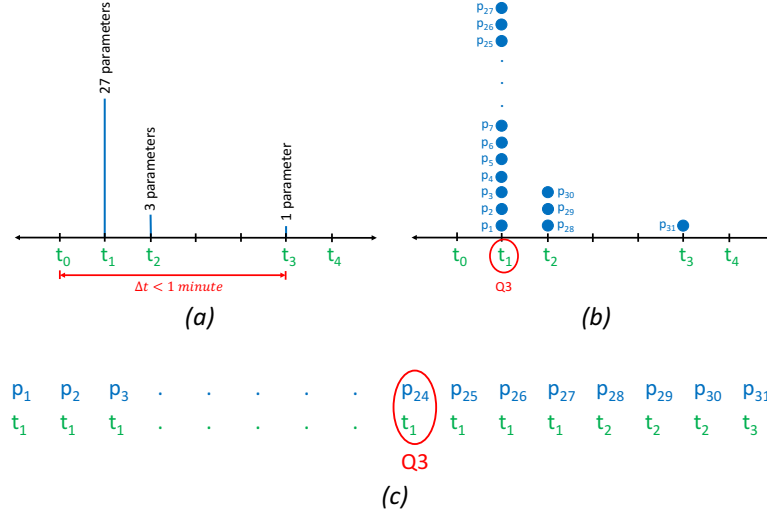


Figure 4.15 Parameter distribution in one polling (example 1).

The formula (17) enables us to calculate the $Q3$ position, where it is understood that n is the number of collected parameters. In the above example, the $Q3$ position has a value of 24. Timestamp t_1 corresponds with the 24th parameter time, so 75% of the data arrive between t_0 and t_1 . According to Figure 4.15, the parameter collection carries out over the polling minute, and the $Q3$ position confirms this. The selected timestamp for the polling $Q_{1 \times 1}$ is t_0 ; in other words, it is t_1 with granularity per minute or the zeroth minute of $Q3$.

$$Q3 = \begin{cases} \frac{3(n+1)}{4} & , n \text{ is an odd number} \\ \frac{3n}{4} & , n \text{ is an even number} \end{cases} \quad (17)$$

Figure 4.16 shows a different example. Suppose a poll of 68 parameters, and the parameters arrive at four different times distributed in an interval higher than or equal to 1 minute. The polling time is t_0 , and the parameters arrive at times t_1, t_2, t_3 , and t_4 . Using the (17) formula, the position of $Q3$ is 51, which means t_4 is the moment when at least 75% of the parameters have arrived. Hence, the polling's assigned timestamp is at the next minute to the polling time and equal to the zeroth minute of t_4 .

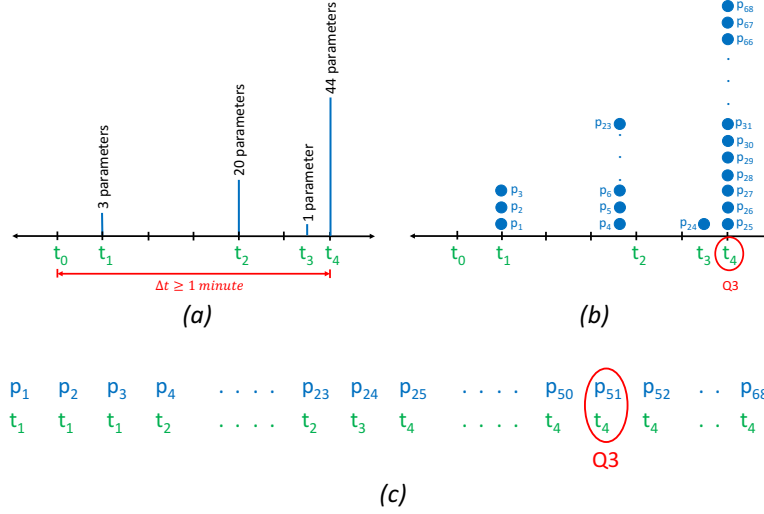


Figure 4.16 Parameter distribution in one polling (example 2).

Additionally, the time-based grouping process generates a new attribute that indicates the distribution range of the arrives of the same polling parameters. It is the time range that exists between the arrival timestamp of the first and last parameter. For example, the range is equal to $t_3 - t_1$ in Figure 4.15, while the range is equal to $t_4 - t_1$, for the Figure 4.16 example. It is a new derived feature of the SOFI dataset (see equation (18)). This is why the size of the resulting matrix D is $m \times (v + 1)$ as (12) defines.

$$Y_{1 \times 1} = \text{range of the arrives of the parameters from the same polling} \quad (18)$$

2.1.2 Interface-based grouping

SOFI raw dataset has a dimension proportional to the number of active interfaces in the peripheral device, because each parameter related to both network traffic and interface status, are measured for every interface. Consequently, this work proposes to standardize the size of the dataset by grouping interfaces and, therefore, the parameters measured in them.

As Figure 4.17 shows, we propose to divide interfaces of peripheral devices into three levels. The internal level, which connects the device with the monitored network. The external level, in charge of connecting the device with external networks. Finally, the peripheral level, which connects the device with other elements at the same level. Each level represents an interface set determined by the statements (19) to (22).

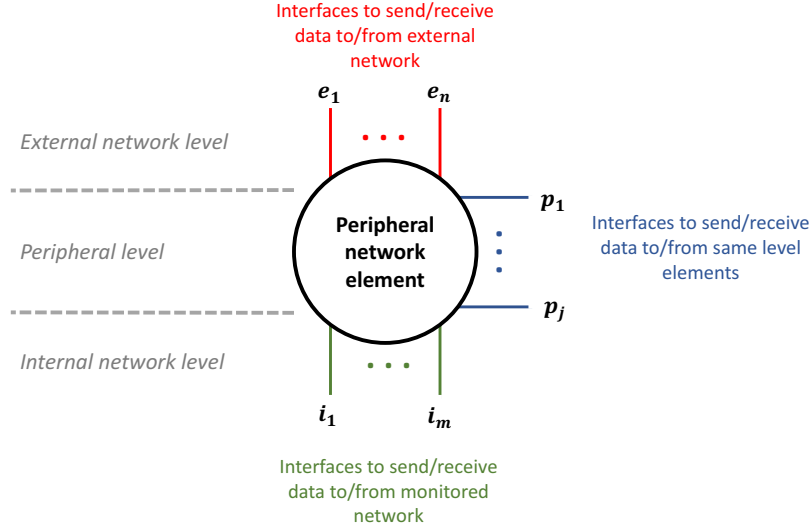


Figure 4.17 Interface grouping scheme.

$$I = \{i \mid i \text{ is a network interface that sends/receives data from the monitored network}\} \quad (19)$$

$$car(I) = x$$

$$E = \{e \mid e \text{ is a network interface that send/receives data from the external network}\} \quad (20)$$

$$car(E) = y$$

$$P = \{p \mid p \text{ is a network interface that sends/receives data from same level elements}\} \quad (21)$$

$$car(P) = z$$

$$A = \{a \mid a \text{ is a network interface of the peripheral device}\} \quad (22)$$

$$I \in A, \quad E \in A, \quad P \in A$$

$$A = I \cup E \cup P$$

$$car(A) = x + y + z$$

As mentioned in section 1.1.3 of this chapter, the raw dataset consists of SNMP parameters classified into three types: traffic descriptors, status descriptors, and identification descriptors. The traffic descriptors are nine parameters measured in the interfaces; therefore, they must be grouped by interface sets. So, for each interface set (I , E , and P), the values as traffic descriptors must be calculated based on the parameters of the individual interfaces. For this purpose, we divide the traffic descriptors into three subtypes and define an operation for each subtype.

First, those that measure an aspect of interface behavior must be summed. Equation (23) declares this sum, where B is a behavior parameter for any interface set (I , E , and P), β is a behavior parameter measured on an interface of the corresponding set, and j is the set size.

Second, if the parameter describes an aspect related to the interface type, its value for each interface set is equal to the measure on any interface of the set. Equation (24) shows this declaration, where A is a parameter of type, for any interface set (I , E , and P), α is a parameter of type measured on an interface of the corresponding set, and j is the set size.

Third, those that measure the interface status with binary values must be averaged. Equation (25) defines this average, where Γ is a status parameter for any interface set (I , E , and P), γ is a behavior parameter measured on an interface of the corresponding set, and j is the set size.

$$B = \sum_{j=1}^j \beta_j \quad (23)$$

$$A = \alpha_j \quad (24)$$

$$\Gamma = \frac{\sum_{j=1}^j \gamma_j}{j} \quad (25)$$

Table 4.6 details the subtype of each parameter, the description of each subtype, and the operation to be performed for the grouping.

Table 4.6 Operations for each subtype of traffic descriptors.

PARAMETER	SUBTYPE	DESCRIPTION	OPERATION
Bits_received	Behavior	They are parameters that describe the behavior of the interfaces.	(23)
Bits_sent			
Inbound_packets_discarded			
Inbound_packets_with_errors			
Outbound_packets_discarded			
Outbound_packets_with_errors			
Interface_type	Type	They are parameters that describe the interface type.	(24)
Speed			
Operational_status	Status	They are parameters that determine the status of the interfaces.	(25)

All in all, the operations reduce the multiple traffic descriptors to nine parameters for each interface set, that is, twenty-seven descriptors added to five status descriptors and one derived descriptor, (see Table B.3 of Annex B), for a total of thirty-three fixed parameters. The collected five identification descriptors measured in the device (Device_contact_details, Device_description, Device_location, Device_name, and System_object_ID) identify each peripheral element and do not provide relevant knowledge, so they were not used.

In other words, the result of operations is a matrix S of thirty-four columns, including the timestamp attribute, as (26) defines.

$$S_{m \times n} = [s_{ab}]_{1 \leq b \leq n}^{1 \leq a \leq m}; n = 34 \quad (26)$$

2.1.3 Event correlation or labeling

We considered two ways to label the dataset, but only one was selected. The first one considers the timestamps when the device that causes the failure triggers events during induction. The second one considers the time interval of the well-known induced faults. As mentioned in section 1.2.2 of this chapter, there is a delay between the moment of fault induction and the events generated due

to it, this condition evidences that the events are generated long after the network is already failing. Therefore, they cannot be considered to detect the failure in a timely manner. For this reason, this work labels the SOFI dataset through a well-known failure set.

As mentioned in the data collection experiment (section 1.1.2 of this chapter), we register the type, occurrence layer, and time interval for each induced failure. So, the induced failure set can be grouped into several subsets according to the occurrence layer. As Figure 4.18(a) shows, there are n layers below the peripheral layer, then there will be n failure subsets. In this particular case, the access layer and distribution layer are the deeper layers of the monitored network; thus, there are two subsets to bear in mind.

The labeling process takes each failure subset to label the SOFI dataset. As a consequence, the labeling result is a multilabel dataset, as shown in Figure 4.18(b). This means that there is a labeled SOFI dataset for each monitored layer and that a SOFI dataset instance could have one or two classes.

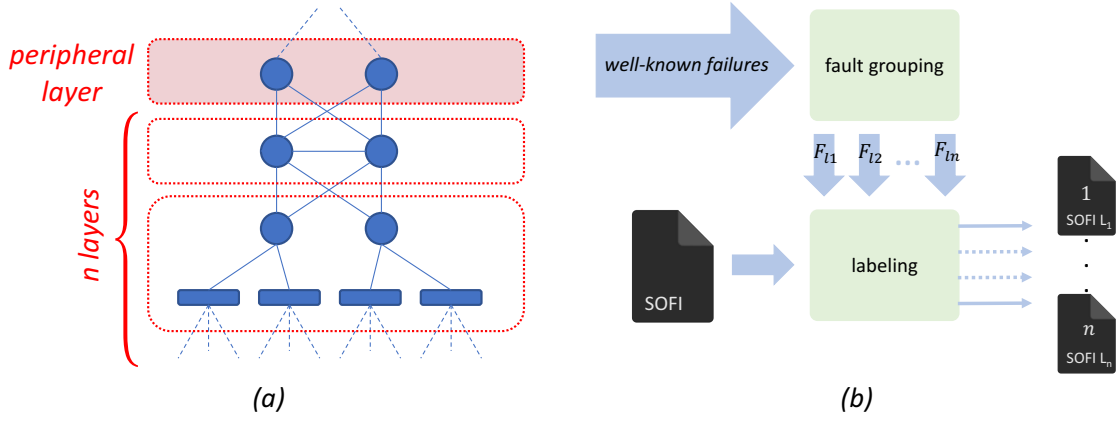


Figure 4.18 SOFI labeling approach.

The labeling process finds, for each well-known failure, the instances whose timestamps comply with the condition (27) and label them according to the layer. A if the failure origin is in the access layer. D if the cause is in the distribution layer. If an instance does not meet the condition for any failure, its label is NE .

$$timestampFault_i \leq timestampInstance \leq timetampFault_f \quad (27)$$

By joining the two labels, we have a data set that identifies if a failure occurs in an internal layer (F label) or if a failure does not occur (NE label). It is important to note that this work does not consider failures that overlap in time at the same level or different levels. Nor does it use a high granularity in labeling. We only define whether or not there is a fault, but we do not differentiate the type of failure or its cause. These two issues are future challenges.

3 SOFI dataset summary

The SOFI (Symptom-Fault relationship for IP-Network) dataset is the result of the implementation of the previous processes in a monitored network of two internal levels. SOFI was collected at a

different time in several days. It contains **12971 instances**, which correspond to about 649 hours of network monitoring, and during around 10 hours of that time, the network was fault induced at different time intervals.

SOFI has three labeled versions. All versions have the same instances and are differentiated by the label classes:

- ✓ The first version is labeled by NE and F classes, where NE indicates normal behavior, and F represents in failure behavior of the monitored network regardless of the layer where the failure occurs. This dataset version is the object of study.
- ✓ The second version is labeled by NE and A classes, where A indicates in failure behavior in the access layer. This version does not contain failure information about the distribution layer.
- ✓ The third version is labeled by NE and D, where D indicates in failure behavior in the distribution layer. This third version does not contain failure information about the distribution layer.

As evidenced by the relationship of hours of monitoring in normal and in failure states, SOFI is a highly imbalanced dataset with an approximate **1:70 ratio**, being failure class the minority class.

Finally, each sample in the dataset consists of the **33 features**⁶ listed in Table 4.7 and the timestamp and a label class.

Table 4.7 SOFI attributes list.

SOFI ATTRIBUTES		
P_Bits_received	EX_Inbound_packets_with_errors	IN_Interface_type
IN_Bits_received	P_Outbound_packets_discarded	EX_Interface_type
EX_Bits_received	IN_Outbound_packets_discarded	P_Speed
P_Bits_sent	EX_Outbound_packets_discarded	IN_Speed
IN_Bits_sent	P_Outbound_packets_with_errors	EX_Speed
EX_Bits_sent	IN_Outbound_packets_with_errors	SNMP_availability
P_Inbound_packets_discarded	EX_Outbound_packets_with_errors	ICMP_loss
IN_Inbound_packets_discarded	P_Operational_status	ICMP_ping
EX_Inbound_packets_discarded	IN_Operational_status	ICMP_response_time
P_Inbound_packets_with_errors	EX_Operational_status	Device_uptime
IN_Inbound_packets_with_errors	P_Interface_type	Range

SOFI repository is <https://data.mendeley.com/datasets/tc6ysmh5j8/> (Vargas-Arcila et al., 2021).

⁶ The collected identification descriptors like Device_name, Device_description, and so on, were not used because they are identifying data for each peripheral element and do not provide relevant knowledge. The collected raw data are described Table B.3 in Annex B.

Chapter V: Experiments

*Every stumble is not a fall,
and every fall does not mean failure*
(Oprah Winfrey)

This chapter presents the experiments and analyses that support the viability of our Peripheral Fault Diagnosis Model for IP-based Networks (PALADIN). The first section corresponds to a data exploring that allows us to reveal the existence of the fault propagation phenomenon to a peripheral level, which demonstrates the feasibility of our approach. The second section compares several incremental learning algorithms for data streams implemented according to the components of the diagnosis model. This same section compares the performance of these algorithms with the results of their basic implementation. Both explorations suggest sufficient pieces of evidence to support our proposal.

1 Structured data exploring

If there is a sudden change in the behavior of the monitored parameters in a peripheral device when the failure occurs in an internal element, it would mean that there is a fault propagation phenomenon. So, those parameter changes would represent the failure symptoms. This is why we observed the behavior of all the monitored parameters for several failure time windows, including pre- and post-failure time intervals. We certainly use the SOFI dataset for the analysis.

There are four findings as a result of observations. Table 5.1 indicates the parameters related to each finding. Firstly, a parameter subset has constant values not only at failure time but also at all times. This parameter subset contains three status descriptors and seven traffic descriptors. Within the traffic descriptors, two are those which determine the interfaces type, one is the parameter which determines the operational status of the interfaces, and four that measure the packets discarded and with errors in the interfaces.

Table 5.1 Parameter's findings details.

FINDING	PARAMETER		
	NAME	INTERFACES	TYPE
CONSTANT BEHAVIOR	Inbound_packets_discarded	P, I, E	Traffic descriptor (Behavior)
	Inbound_packets_with_errors	P	
	Outbound_packets_discarded	P, I, E	

<i>(not only at failure time but also at all times)</i>	Outbound_packets_with_errors	P	
	Operational_status	P, I, E	Traffic descriptor (Status)
	Interface_type	P, I, E	Traffic descriptor (Type)
	Speed	P, I, E	
	SNMP_availability	N/A	Status descriptor
	ICMP_loss	N/A	
	ICMP_ping	N/A	
<i>SPORADIC PEAKS</i>	Range	N/A	Calculated parameter
<i>VARIABLE BEHAVIOR without significant change</i>	Bits_received	P	Traffic descriptor (Behavior)
	Bits_sent	P	
	Inbound_packets_with_errors	I, E	
	Outbound_packets_with_errors	I, E	
	ICMP_response_time	N/A	Status descriptor
	Device_uptime	N/A	
<i>AFFECTED BEHAVIOR when the failure occurs</i>	Bits_received	I, E	Traffic descriptor
	Bits_sent	I, E	

Secondly, the Range parameter has a constant value at failure time but presents sporadic peaks at different timestamp outside failures. Figure 5.1 shows the Range parameter behavior at two ample time intervals for the two peripheral elements (SwitchCore I and SwitchCore II). The plotted time intervals cover different failure times. Although the sudden changes of the Range parameter do not match those times, these could be due to a change in the network traffic behavior (increase or decrease in traffic), so Range is not a parameter directly related to failures.

Thirdly, several parameters behave as variables without a notably significant change at failure time. Figure 5.2 to Figure 5.5 show the behavior of five of them for the two peripheral elements and the failure types under the spotlight. We have selected two impact failures to show; the first is "Distribution layer going down", caused in the access layer, and the second is "protocol issues", caused in the distribution layer. The plots illustrate the failure time window broadened to note the parameter behavior before, during (red timeline), and after failure.

As figures show, the Device_uptime descriptor presents a linear behavior (Figure 5.4), the Inbound_packets_with_errors parameter for the external interfaces presents sporadic peaks even sometimes within the failures time window (Figure 5.5), and the other attributes show variable values. Hence, there is no pattern of behavior change for any of these parameters, so it is not possible to determine at a glance if they are affected or not during a failure.

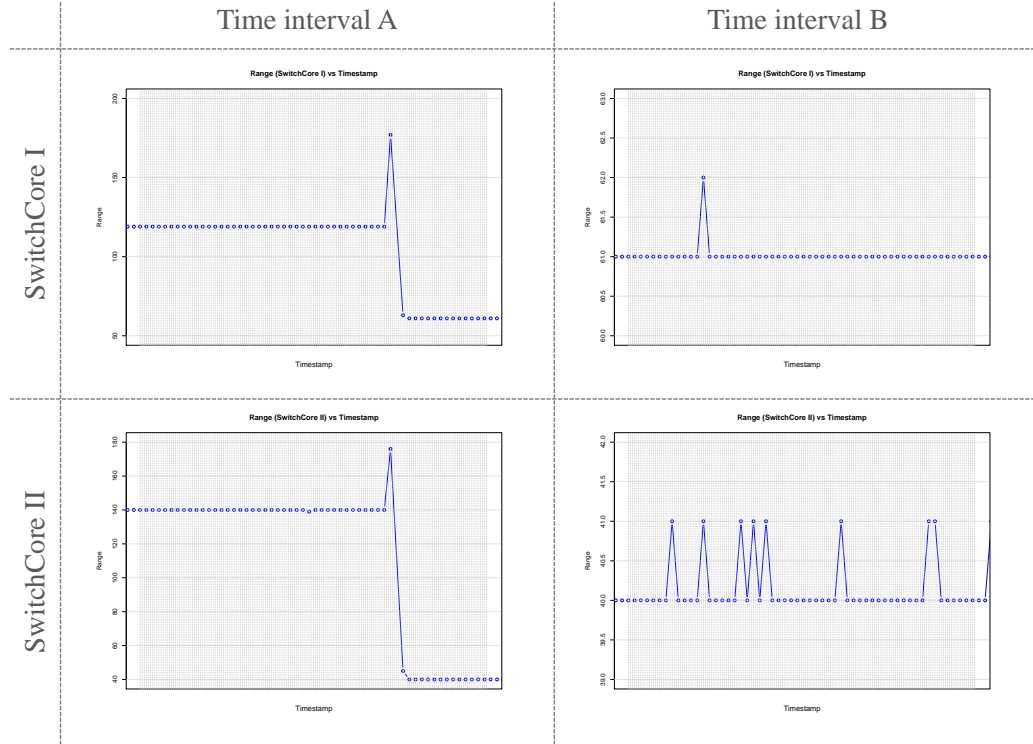


Figure 5.1 Range parameter behavior at different time intervals.



Figure 5.2 ICMP_response_time parameter behavior before, during, and after the failure. The failure time interval is indicated in red.



Figure 5.3 *P_Bits_received* and *P_Bits_sent* parameters behavior before, during, and after the failure. The failure time interval is indicated in red.

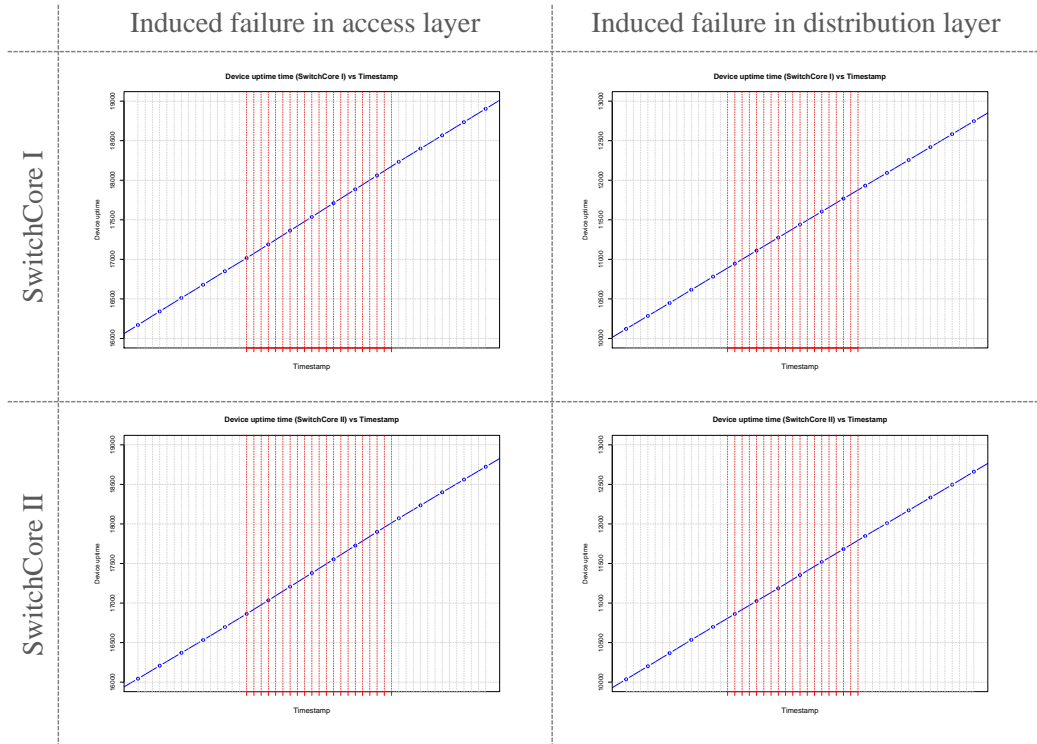


Figure 5.4 *Device_uptime* parameter behavior before, during, and after the failure. The failure time interval is indicated in red.



Figure 5.5 *E_inbound_packets_with_errors* parameter behavior before, during, and after the failure. The failure time interval is indicated in red.

Fourthly, the parameters *Bits_received* (for I and E interface set) and *Bits_sent* (for I and E interface set) change their values due to the induced failures, as Figure 5.6 shows. For example, the parameter *Bits_received* for the external interface set decreases its value at the polling time immediately after causing a failure. Meanwhile, the *Bits_received* discriminator for the internal interface set presents a significant decrease or increase (depending on the peripheral device and the induced failure) once the failure is induced. Likewise, the four parameters have a recovery time after the failure window.

As Figure 5.6 well demonstrates, these parameters are affected when the failure occurs; therefore, we interpret them as failure symptoms as well as evidence of the existence of the fault propagation phenomenon, which makes feasible the hypothesis of this research.

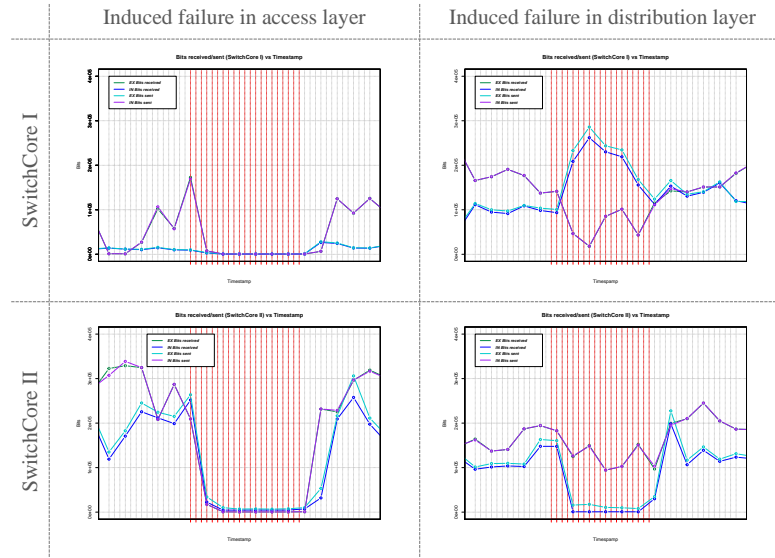


Figure 5.6 *E_Bits_received*, *I_Bits_received*, *E_Bits_sent* and *I_Bits_sent* parameters behavior before, during, and after the failure. The failure time interval is indicated in red.

The previous observations suggest that a fault propagation phenomenon occurs when a fault takes place in internal levels of a monitored network. This phenomenon only happens for impact failures, and its manifestation is the change in behavior of some parameters monitored from the peripheral level.

Parameters that manifest the failure symptoms are those related to the traffic carried by the peripheral devices. Then, they present either a gradual increase or decrease in their value during the failure time. This gradual behavior is also attributed to fault propagation. Likewise, the recovery values of each affected parameter are evident, so we affirm that as there is a propagation time, there is also a recovery time of the failure, both reflected in the level of observation of the symptoms.

In conclusion, the proposed peripheral and non-invasive monitoring approach is feasible for the diagnosis of impact link failures. It represents a novelty way with the possibility of solving persistent gaps in traditional works.

2 A brief experimental analysis for traditional machine learning models with the SOFI dataset

In order to have a deeper knowledge base about the collected and labeled data (SOFI dataset), this section analyses the fault classification performance of several traditional machine learning algorithms and applies resampling methods to deal with imbalanced data.

2.1 Experimental setup

Using the Weka tool's experimenter environment (Witten et al., 2017), an experiment was configured for testing and training seven traditional machine learning algorithms over the SOFI dataset⁷. Those algorithms are Naive Bayes, Multilayer Perceptron, K-Nearest Neighbors, Boosting, Bagging, C4.5, and PART.

It should be pointed out that the SOFI dataset consists of two dataset files because there are two peripheral network elements, so both collect network monitoring data. All the dataset files' values were normalized and resampled to get two new SOFI versions for each peripheral element. This was done because we already know that SOFI is an imbalanced dataset, so it is necessary to consider techniques that avoid algorithms ignoring the learning from the minority class or instances that represent network failures. So, two resampling methods were used: undersampling and oversampling.

The undersampling approach reduces the dataset size by eliminating several instances belonging to the majority class and preserving all the samples belonging to the minority class. This experiment

⁷ It is important to clarify that the timestamp attribute was not included in this experiment since this parameter only indicates the instances' order.

implements random undersampling, that is, random deletion from the healthy network instances to obtain a small subset of the majority class and a 1:1 ratio of balancing.

The oversampling method creates new instances from the minority class by synthetic creation or replicating samples. This experiment implements Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) to get synthetic instances from in fault network state. SMOTE performs an iterative process which can be explained using Figure 5.7. First, it randomly selects an instance from the minority class (vector \vec{A}) to get the nearest neighbor (vector \vec{B}), and it takes the difference between the two ($\vec{A} - \vec{B}$ line). Then, it multiplies the difference with a random number between 0 and 1. The above finds a random point (p) along the line between the vectors and therefore causes a new synthetic instance (\vec{P}) (Fernández et al., 2018b).

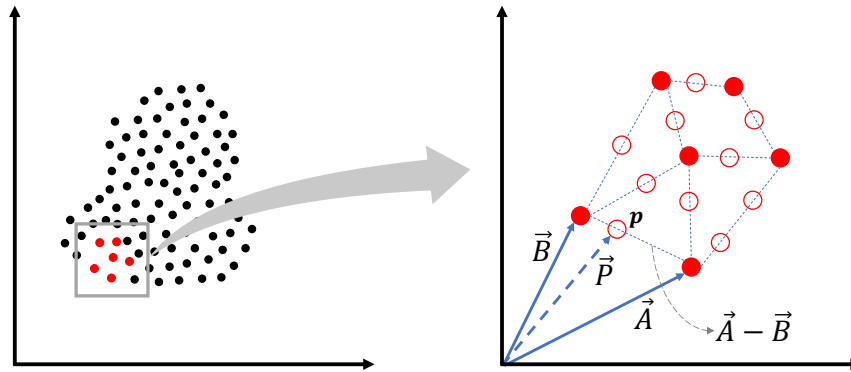


Figure 5.7 Synthetic Minority Oversampling Technique (SMOTE).

In summary, the Weka experiment executed seven algorithms over three different SOFI versions (original, undersampled, oversampled), and each version consists of two dataset files.

The five evaluation metrics described in the next section were taken as references to compare the algorithms' performance and select the best models. Furthermore, the popular k -fold cross-validation method was configured to estimate the metrics values. The cross-validation involves splitting the dataset into k groups or folds, and the model training and testing are executed k times. The first time, the first group is the testing data, while the model fits the other folds. The second time, the second group is the testing data, while the model is fitted on the other folds, and so on. At the end of iterations, the performance is estimated by averaging the results from the k executions (Zhang et al., 2020).

2.2 Performance evaluation metrics

The most used and easy way to evaluate the classification models is computing the accuracy (percentage of correctly classified instances). Nevertheless, as has been so often repeated, the network fault classification is a severely class imbalance problem where the minority class corresponds to on failure network state. As an assumption, if all the healthy states of the network were classified correctly but the failure states were not, a high precision would be obtained. Then using accuracy is a mistake. Besides, misclassifying an on failure state is costlier for network management than determining that a healthy state is a failure.

Hence, different performance metrics are required to evaluate the models' behavior with the minority class mainly, without neglecting to know what happens with the majority class.

Thus, on the one hand, metrics biased toward one class are used, specifically toward the minority because it is the interest class. Those metrics are the sensitivity and F1-score. Through them, we are going to focus on analyzing the performance of the relevant class classification. On the other hand, metrics that consider errors and correct classification for both classes were used. Those metrics are G-mean, Kappa statistic, and Matthews Correlation Coefficient (MCC). These last are among the most popular for performance evaluation in imbalanced scenarios.

The mentioned five metrics could be better understood by considering the confusion matrix because they are based on its elements. As figure Figure 5.8 represents, the confusion matrix collects the classification results discriminated by classes. There are a row and a column for each class; furthermore, the rows contain the actual class results while the columns are the predicted class. The main diagonal depicts the good results, while the misclassifications are counted off the main diagonal (Witten et al., 2017).

Consequently, a two-dimensional confusion matrix for our fault classification environment contains four elements:

- True Positive (TP): The number of predicted instances as failures and they actually are failures.
- False Positive (FP): The number of predicted instances as failures, but they actually are not.
- False Negative (FN): The number of predicted instances as healthy network class, but they actually are failures.
- True Negative (TN): The number of predicted instances as healthy network class and they actually are healthy states.

		Actual values		
		Failure	Healthy	total
Prediction outcome	Failure	TP	FP	Predicted failures
	Healthy	FN	TN	Predicted healthy states
total		Actual failures	Actual healthy states	

Figure 5.8 Confusion matrix structure.

Below is a detailed description of each metric:

- **Sensitivity / true positive rate (TPR) / recall**

The sensitivity is the classification accuracy of the relevant class. Using the confusion matrix terminology, it represents how well a model can identify the true positives (Ting, 2010b), and it is also known as recall or true positive rate (TPR) (Sammur & Webb, 2010b). Equation (28) indicates how to calculate the sensitivity.

$$Sensitivity = \frac{TP}{TP + FN} \quad (28)$$

In this context, the relevant class indicates on failure network state, so the sensitivity is the fraction of faults predicted correctly by a model.

- **F1 Score**

In mathematical terms, the F1 measure is the harmonic mean of precision and recall. The precision is the proportion of true positives of the total number of positives predicted (Ting, 2010a). Those metrics' equations, (28) and (29), reflect that the F1 measure focuses on the analysis of the relevant class, specifically in assess how appropriate are the predictions as the relevant class (Fernández et al., 2018b), but also assumes the false negatives and false positives are equally costly. Equation (30) defines how to compute it.

$$Precision = \frac{TP}{TP + FP} \quad (29)$$

$$F1 = 2 \frac{Precision \times Recall}{Precision + Recall} \quad (30)$$

- **Geometric mean (G-Mean)**

In mathematical terms, the geometric mean is the real positive n -th root of n numbers (Clark-Carter, 2010). This work implements the geometric mean of sensitivity and specificity as (Kubat et al., 1997) propose (equation (31)). Thus, this metric considers the accuracies of both majority and minority classes separately, so it is independent of the class distribution and insensitive to class imbalance issues (Kubat et al., 1998; S. Wang et al., 2013).

$$G_{Mean} = \sqrt{sensitivity \times specificity} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \quad (31)$$

- **Kappa statistic**

The coefficient K was introduced by (Cohen, 1960), and it is best known as Cohen's kappa statistic. Equation (32) defines it considering two relevant values, the proportion of instances correctly classified (p_0) and the proportion of instances for which the classification is expected by chance (p_c).

As may be seen, this coefficient compensates the accuracy of a classifier considering the proportion of successful classifications that could be obtained at random (Fernández et al., 2018b). Thus, kappa statistic is generally preferred over the traditional accuracy (equivalent to p_0). p_0 and p_c can be calculated with the confusion matrix's aid as the equations (33) and (34) define.

$$K = \frac{p_0 - p_c}{1 - p_c} \quad (32)$$

$$p_0 = \frac{TP + TN}{n} \quad (33)$$

$$p_c = \frac{(TP + FN) \times (TP + FP)}{n^2} + \frac{(FP + TN) \times (FN + TN)}{n^2} \quad (34)$$

- **Matthews correlation coefficient (MCC)**

The MCC metric was introduced by (Matthews, 1975) for performance evaluation in biochemical environments. It is widely used for class imbalance contexts. Its formula arises from the Pearson's correlation coefficient for binary vectors, so it is known as the discretization of Pearson's correlation coefficient (Boughorbel et al., 2017) for binary classification.

In other words, MCC indicates the correlation between observed and predicted classes, and it can be expressed in terms of all confusion matrix elements, as equation (35) defines. It is widely used for class imbalance contexts because it gives equal importance to both classes regardless of their skew (Boughorbel et al., 2017).

$$MCC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN)(TP + FP)(FP + TN)(FN + TN)}} \quad (35)$$

Its value ranges from -1 to 1. -1 indicates a negative correlation, so the classification is wholly wrong. 0 means no correlation, so the classification is no better than random. 1 denotes a complete correlation, so the classification is always correct, then a value closer to 1 is preferable (Fernández et al., 2018b).

2.3 Experimental results and discussion

A paired T-tester executed in Weka compares the results delivered by the same tool. This way, it is possible to know the statistical significance, that is, if an algorithm is statistically better or worse than another. The Bagging algorithm is selected as the test base to perform the T-test because it has one of the best performances, carrying out a fast revision.

Table 5.2 to Table 5.7 contain the metrics results. The "*" and "v" characters refer to a statistically significant difference from the test base; "*" when the results are lower than the base algorithm, and "v" when the results are larger. Weka provides the statistical significance for sensitivity, kappa, and MCC, so the tables indicate it. Nevertheless, F1-score and G-Mean are not provided by Weka; they were computed using the confusion matrix data, so their statistical significance is not available. Tables also highlight the column of the base algorithm in another color.

Before analyzing the tables' results, it is important to note that the specificity or true negative rate metric was verified for all the tests carried out. This metric focuses on evaluating the majority class's learning, and always its results are close to 1 (range between 0.81 and 0.99). This means that both with the original dataset and the balanced ones, the algorithms never neglect the majority class.

Now, Table 5.2 and Table 5.3 provide the results with the original SOFI dataset. As shown, none of the algorithms return good results, and they cannot learn from the minority class. All metrics are poor, especially with Naive Bayes, Multilayer Perceptron, and KNN algorithms. On the other hand, Boosting, Bagging, C2.5, and PART have results slightly less bad. Nevertheless, none is statistically better than the other, and the results are not sufficient to conclude a good performance.

Table 5.2 Classification performance with original SOFI dataset (SwitchCore-I file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,307	0,001 *	0,311	0,447	0,390	0,403	0,390
F1 score	0,084	0,003	0,358	0,574	0,529	0,529	0,513
G-Mean	0,529	0,038	0,556	0,668	0,624	0,634	0,624
Kappa	0,060 *	0,000 *	0,340 *	0,560	0,510	0,510	0,500
MCC	0,090 *	0,005 *	0,354 *	0,594	0,562	0,552	0,535

Table 5.3 Classification performance with original SOFI dataset (SwitchCore-II file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,329	0,011 *	0,317	0,436	0,399	0,382	0,384
F1 score	0,072	0,021	0,340	0,559	0,537	0,525	0,510
G-Mean	0,538	0,105	0,561	0,660	0,631	0,618	0,619
Kappa	0,050 *	0,020 *	0,330 *	0,540	0,520	0,510	0,490
MCC	0,078 *	0,036 *	0,331 *	0,579	0,568	0,561	0,535

The above is a logical behavior due to the well-known class imbalance issue. There will always be an imbalance situation in the network fault diagnosis context, so it is necessary to apply class rebalancing strategies if acceptable results are to be achieved.

The results in Table 5.4 and Table 5.5 correspond to the executions over the undersampled SOFI dataset version. Although there is a performance improvement over the above tables, Naive Bayes, Multilayer Perceptron, and KNN are still the worst with mediocre responses and significantly different. The other algorithms have good behavior with the minority class, as reflected by the sensitivity and F1-score metrics (above 0,8). Based on the G-Mean, Kappa, and MCC, these algorithms also have a good balance between observed and predicted classes, though they do not exceed the 0,7 value.

On the other hand, an important consideration is that the evaluation is performed with cross-validation with a very small SOFI subset, due to undersampling. So, these results possibly are due to the overfitting of the model.

Table 5.4 Classification performance with undersampled SOFI dataset (SwitchCore-I file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,785	0,790	0,835	0,850	0,815	0,834	0,818
F1 score	0,725	0,734	0,794	0,846	0,833	0,833	0,822
G-Mean	0,697	0,710	0,782	0,846	0,836	0,832	0,822
Kappa	0,400 *	0,430 *	0,570	0,690	0,670	0,660	0,640
MCC	0,410 *	0,432 *	0,571 *	0,691	0,673	0,665	0,645

Table 5.5 Classification performance with undersampled SOFI dataset (SwitchCore-II file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,782	0,803	0,804	0,857	0,843	0,819	0,839
F1 score	0,727	0,753	0,790	0,851	0,849	0,832	0,852

G-Mean	0,729	0,754	0,790	0,851	0,849	0,832	0,853
Kappa	0,410 *	0,470 *	0,570 *	0,700	0,700	0,670	0,710
MCC	0,417 *	0,476 *	0,573	0,699	0,699	0,670	0,710

Table 5.6 and Table 5.7 indicate the classification results with the oversampling version of the SOFI dataset. As shown clearly, the algorithms have excellent results except for Naive Bayes. All the metrics are high (near to 1), so it is necessary to analyze the significant differences between them.

If the sensitivity is analyzed, Boosting is significantly better than the others for both dataset files. KNN and the test base are not significantly different between them for one of the dataset files, but with the other, KNN is better than Bagging. Moreover, although the other algorithms present high results, they are significantly worse than the test base. So, Boosting creates a better model for minority class classification.

If the significances of kappa and MCC are observed, there are three things to be said. The PART results are not significantly different from Bagging. Boosting is significantly better than the test base. Furthermore, the other algorithms are significantly worst than Boosting, Bagging, and PART. Thus, Boosting creates a better model for both classes.

Table 5.6 Classification performance with oversampled SOFI dataset (SwitchCore-I file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,800 *	0,939 *	0,996	0,998 v	0,995	0,990 *	0,991 *
F1 score	0,713	0,911	0,985	0,996	0,989	0,985	0,988
G-Mean	0,666	0,908	0,985	0,996	0,989	0,985	0,988
Kappa	0,350 *	0,820 *	0,970 *	0,990 v	0,980	0,970 *	0,980
MCC	0,366 *	0,819 *	0,970 *	0,991 v	0,979	0,971 *	0,977

Table 5.7 Classification performance with oversampled SOFI dataset (SwitchCore-II file)

	NB	MP	KNN	Boosting	Bagging	C2.5	PART
Sensitivity	0,756 *	0,965 *	0,997 v	0,997 v	0,994	0,988 *	0,990 *
F1 score	0,705	0,923	0,984	0,995	0,989	0,982	0,989
G-Mean	0,679	0,918	0,983	0,995	0,989	0,982	0,988
Kappa	0,370 *	0,840 *	0,970 *	0,990 v	0,980	0,960 *	0,980
MCC	0,370 *	0,842 *	0,967 *	0,989 v	0,977	0,964 *	0,977

Clearly, the best results are obtained when the data is rebalanced. Now, to analyze which of the two rebalancing approaches is better for the SOFI dataset context, the three test versions' performance has been plotted on a radar chart concerning one of the dataset files (SwitchCore-II).

Figure 5.9 shows three radar charts describing each scenario's performance (original, undersampling, and oversampling). This graphic way allows identifying the differences between the results quickly. The radar's aperture for the oversampling scenario is more remarkable than others because the metrics have better results for almost all the models, and there is no loss of information from the classes. The original and undersampling scenarios do not have a large aperture on the radar, especially the original that tends to concentrate in the graph center, so there is a more significant difference with the ideal behavior. Therefore, the oversampling has thrown better learning outcomes.

What is valuable about this experiment is that it gives us an idea of which resampling method works best with SOFI. This is an important aspect and one which is certainly considered in the diagnosis module of the model proposed. That is, what rebalancing method to incorporate in a data stream classification for network fault classification environments.

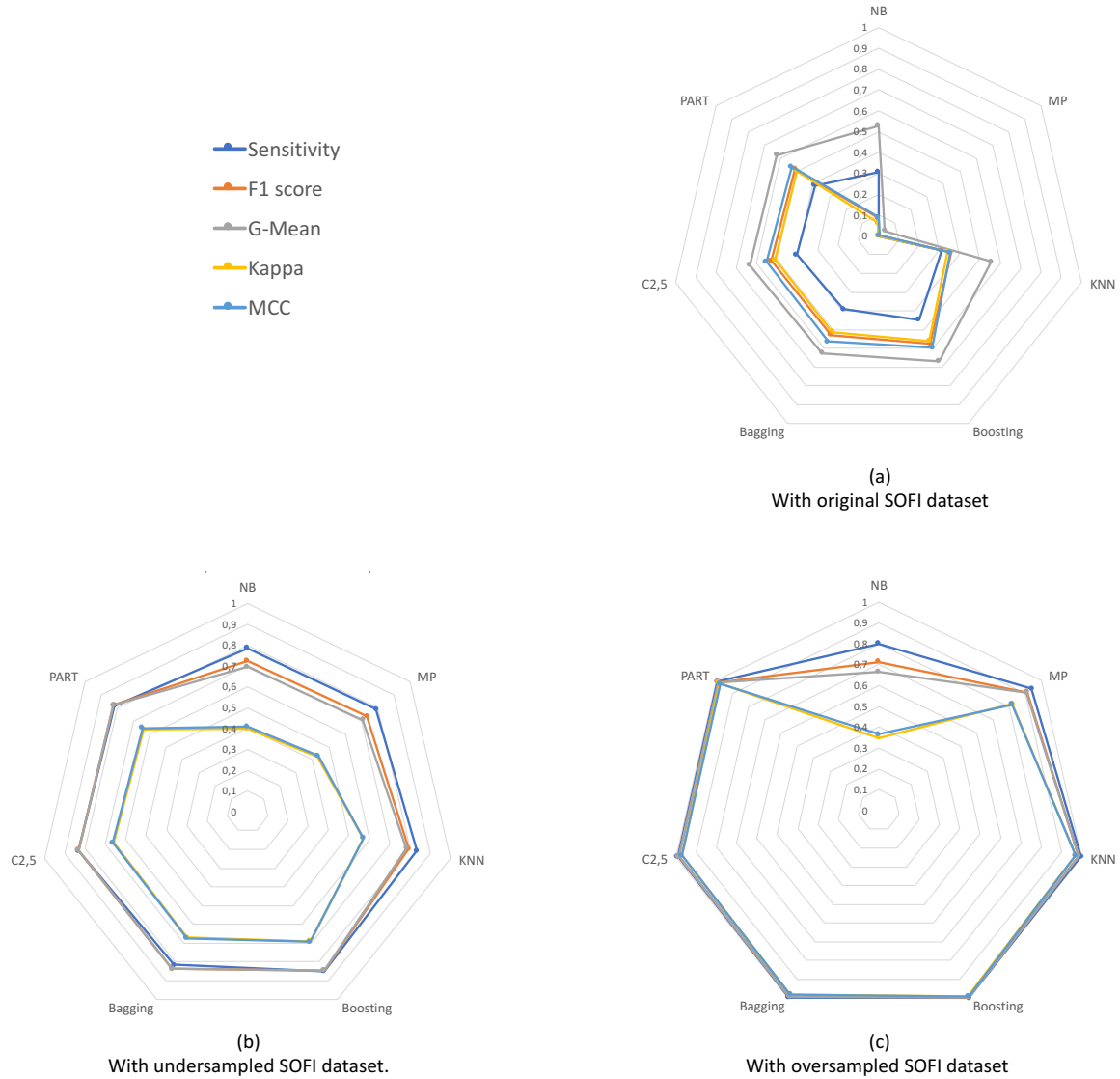


Figure 5.9 Traditional classification performance.

3 PALADIN model evaluation

As mentioned in section 4.2 of Chapter III, the PALADIN model has a Diagnosis module based on three components, a concept-drift detector, a rebalancing method, and an incremental learning algorithm (see Figure 3.10 in Chapter III). This module is a framework to deal with combined difficulties from streaming and imbalanced data. So, this section aims to implement it through 25 different incremental algorithms, the most appropriate rebalance technique according to section 2.3 analysis of the current chapter, and the ADWIN algorithm as the concept-drift detector.

This section also contains an in-depth analysis of the results of this implementation. The test of this implementation constitutes the evaluation of the entire PALADIN model since we use the SOFI dataset⁸, obtained through the first three PALADIN modules, as the incoming data stream to the last module (Diagnosis module). So, the experiments described here represent the third specific objective of this research.

3.1 Experimental setup

The experiment is based on the meta-strategy proposed by (Bernardo et al., 2019) called RebalanceStream, which can rebalance a data stream and train a model. We can frame this strategy within the Diagnosis module of the current work because it comprises the proposed diagnosis components. RebalanceStream uses ADWIN to detect if a concept-drift is in the stream. When that detection is successful, it rebalances the data arrived until then through SMOTE. Then, it trains a new model using the rebalanced data. The two models (the current and the new one) are evaluated through the Kappa-statistic performance evaluation methodology (Bifet et al., 2015). The best trained model is chosen to continue. Figure 5.10 shows the flowchart of the process.

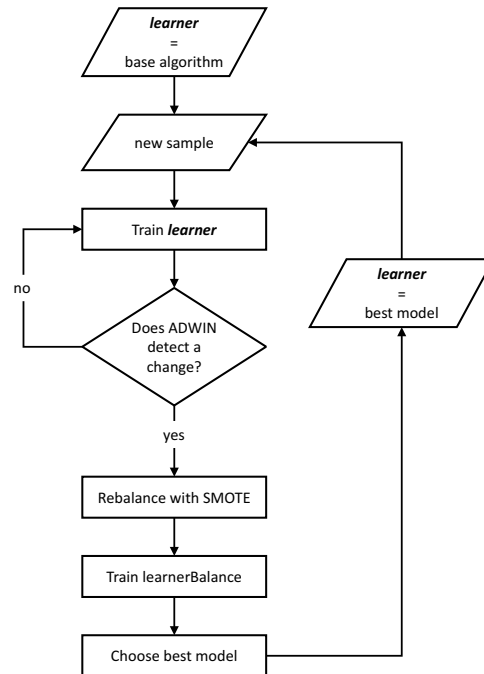


Figure 5.10 Flowchart of RebalanceStream meta-strategy (Bernardo, 2019).

The work (Bernardo et al., 2019) evidently has PALADIN's diagnosis module components, so it provides the basis for our evaluation, and its code has been reused. Nevertheless, that code only receives a synthetic data stream, is configured for one single online algorithm and provides performance results based on a single metric. So, the code was modified to support any real data

⁸ It is important to clarify that the timestamp attribute was not included in this experiment since this parameter only indicates the instances' arriving order.

stream and throw four additional performance measures. The implementation was written in the Java language and using the MOA API (Bifet, Holmes, Kirkby, et al., 2010).

The elements used in each component of our Diagnostic module are described in detail below. The implementation of each element preserves the default parameters that MOA has configured.

- **Concept-drift detector and class imbalance detector:** ADaptive sliding WINdow (ADWIN) (Bifet & Gavalda, 2007) is a change detection method that uses a variable-length window of recent instances for monitoring the prediction errors for those instances. Suppose the window of predictions in Figure 5.11 for a simplified explanation of this algorithm. Number 1 indicates correct prediction, and 0 indicates incorrect prediction. This window W is partitioned into w_0 and w_1 subwindows. w_0 represents the most recent predictions and w_1 the older. For each partition, ADWIN calculates their mean error rate u_0 and u_1 and compares the difference absolute to a given threshold. When the difference is higher than the threshold, a drift is detected. Then, the oldest data of the window (w_1) are dropped.

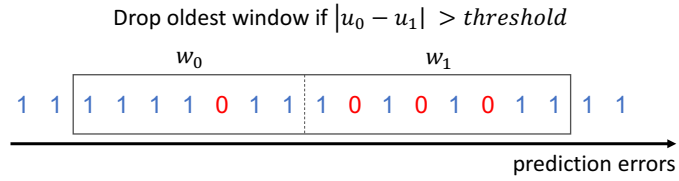


Figure 5.11 How the ADWIN drif-detector works.

ADWIN is widely embedded by incremental algorithms such as CVFDT_{NBC} (Bifet, Frank, et al., 2010), Hoeffding Adaptive Tree (Bifet & Gavalda, 2009), Leveraging Bagging (Bifet, Holmes, & Pfahringer, 2010) and Adaptable Diversity-based Online Boosting (Santos et al., 2014), which validates its great utility for the data stream analysis. Therefore, it is justified to use it in our experiment. So, although there are many other drift detectors (R. Barros & Santos, 2018), ADWIN is sufficient for the research scope.

On the other hand, as mentioned earlier, class imbalance in a data stream is a problem generally associated with concept-drift, so in this experiment, our concept-drift detector and class imbalance detector are both the same algorithm.

- **Rebalancing method:** According to the previous experiment, it is clear that both an undersampling and oversampling technique considerably improve the performance of traditional machine learning algorithms, with the most noted one being the oversampling approach. Based on these results, we will incorporate the SMOTE oversampling technique in this experiment to achieve incremental rebalancing learning.
- **Online classifier:** To have a wide range of data to compare, we choose to implement the framework for 25 different incremental algorithms and evaluate if the fault diagnostic module proposed can introduce improvements in the classification process with imbalanced data streams. Table 5.8 specifies in chronological order all the algorithms used through their names, types, and references. As can be seen, we have implemented four classification algorithm types: decision trees, function classifiers, lazy learners, and ensemble methods.

A decision tree algorithm builds a tree structure where each node establishes a split criterion of data based on an attribute's values. So, for classification tasks, each node splits the data into two branches, and each branch of data will reach another decision node that will split them again. This is a recursive process until a leaf of the tree is found. The leaves are the predictors. Decision trees wait to receive enough instances to create reliable statistics and build their structure in an incremental learning approach (Hulten et al., 2001).

Function classifiers are based on linear models but extend to non-linear kernels. Lazy learners keep the instances seen or some of them, and when a new instance arrives, the algorithm makes a prediction based on the closest instances' classes to the incoming one (Bifet et al., 2018).

Ensemble methods are meta-algorithms that combine predictions from smaller models to give one more accurate single prediction. These methods are the most widely used techniques for data stream classification, so this is why they represent the majority in the table list. Work (Gomes, Barddal, et al., 2017) is a comprehensive survey of ensemble methods for data stream classification.

It is noteworthy that several online classifiers embed strategies to deal with concept drift (the first column of Table 5.8 highlights those algorithms). Nevertheless, the independent concept-drift detector of our diagnosis module is still necessary to achieve class rebalancing.

Table 5.8 List of online classifiers for evaluating the PALADIN diagnosis module.

No.	ALGORITHM NAME	TYPE	REFERENCE
1	CVFDT	Decision Trees	(Hulten et al., 2001)
2	Online Bagging	Ensemble method	(Oza & Russell, 2001)
3	Online Boosting	Ensemble method	(Oza & Russell, 2001)
4	Accuracy Weighed Ensemble (AWE)	Ensemble method	(H. Wang et al., 2003)
5	Dynamic Weighted Majority (DWM)	Ensemble method	(Kolter & Maloof, 2007)
6	Hoeffding Option Tree (HOT)	Ensemble method	(Pfahring et al., 2007)
7	Stochastic variant of PEGASOS (Primal Estimated sub-Gradient SOLver for SVM)	Function classifier	(Shalev-Shwartz et al., 2007)
8	CVFDT _{NBC}	Decision Trees	(Bifet, Frank, et al., 2010)
9	Paired learning	Ensemble method	(Bach & Maloof, 2008)
10	ADWIN Bagging	Ensemble method	(Bifet et al., 2009)
11	ASHT (Adaptive-Size Hoeffding Tree) Bagging	Ensemble method	(Bifet et al., 2009)
12	Hoeffding Adaptive Tree	Decision Trees	(Bifet & Gavaldà, 2009)
13	Online Coordinate Boosting	Ensemble method	(Pelossof et al., 2008)
14	Leveraging Bagging	Ensemble method	(Bifet, Holmes, & Pfahring, 2010)
15	ADAGRAD	Function classifier	(Duchi et al., 2011)
16	Learn++.NSE	Ensemble method	(Elwell & Polikar, 2011)
17	OSBoost (Online Smooth Boost)	Ensemble method	(Chen et al., 2012)
18	Accuracy Updated Ensemble (AUE2)	Ensemble method	(Brzezinski & Stefanowski, 2014)
19	Dynamic Adaptation to Concept Changes (DACC)	Ensemble method	(Jaber et al., 2013)
20	Recurring Concept Drifts (RCD)	Ensemble method	(Gonçalves Jr & Barros, 2013)
21	ADOB (Adaptable Diversity-based Online Boosting)	Ensemble method	(Santos et al., 2014)

22	Boosting-like Online Learning Ensemble (BOLE)	Ensemble method	(R. S. M. d Barros et al., 2016)
23	Hoeffding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)	Decision Trees	(Barddal et al., 2016)
24	SAM-kNN (Self Adjusting Memory model for the k Nearest Neighbor)	Lazy learning	(Losing et al., 2016)
25	Adaptive Random Forest (ARF)	Ensemble method	(Gomes, Bifet, et al., 2017)

The SOFI dataset was built through the first modules of the PALADIN model, which means that these data are the incoming data stream of the Diagnosis module, therefore, the ones that were used to evaluate its performance. Since the monitored network used to obtain SOFI contains two peripheral devices, there are two dataset files; one corresponds to the monitoring of CoreSwitch-I and the other of CoreSwitch-II. Each of the 25 versions of the module implemented was executed for the two data sets obtained. It is also important to note that all data were normalized before being processed by the Diagnostic module to avoid memory overflows.

3.2 Evaluation method and imbalanced stream evaluation metrics

As has been seen, the PALADIN Diagnosis module performs an online classification of network failures, so it is a learning model that must be evaluated in a stream context. Prequential evaluation is the most used method to perform this task in that context (Bifet et al., 2015), so we adopt this process.

Figure 5.12 describes the implemented prequential evaluation method. This evaluation uses each incoming instance to test and then train the model, so the evaluation is always done with not seen before instances. Each evaluation process updates the confusion matrix, so we can read the matrix every n incoming instances to compute a set of five performance metrics. It ensures we obtain a history of performance over time, that is, how the fault classification has adapted over time. In this particular experiment, n is equal to 200 samples. For comparison purposes, this evaluation method was applied both for the proposed Diagnostic module that performs incremental rebalancing learning and the online base algorithm without any rebalancing strategy.

We deal with an imbalanced data stream, so traditional metrics are not adequate for the diagnosis module performance evaluation. According to (Fernández et al., 2018b), prequential accuracy is the most used measure, and nevertheless, as discussed before, accuracy is misleading for imbalanced context. So, the same author suggests using, for imbalanced data stream, the prequential AUC, prequential G-mean, and class Recall metrics.

On the other hand, (Bernardo et al., 2019) uses the standard Kappa statistic inside the rebalance stream process, so we will be faithful to the base code and use Kappa statistic as one of the performance measures. Further, the Kappa statistic is widely used when dealing with imbalanced data (Bifet & Frank, 2010). Nevertheless, some studies prefer Mathews Correlation Coefficient (MCC) over the Kappa statistic (Delgado & Tibau, 2019), so it is a measure that will also be taken into account. (Chicco & Jurman, 2020) analyzes the advantages of MCC in binary classification.

The above reasons led us to select these five metrics as the most informative to evaluate the Diagnosis module. The previous section 2 describes most of these metrics, which, applied in a streaming context, are measured following the prequential evaluation process. Therefore, the next item describes in-depth the prequential AUC that has not been explained before.

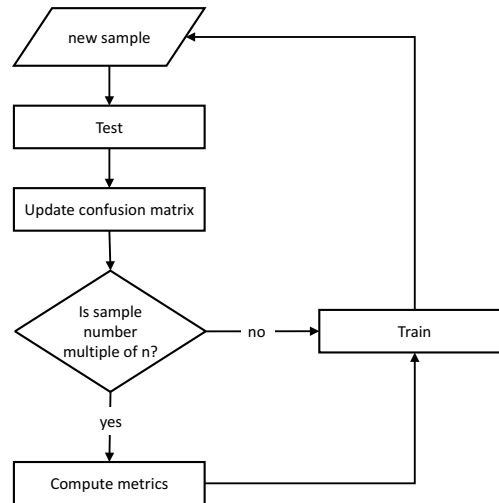


Figure 5.12 Prequential evaluation process.

- **Prequential AUC**

Prequential AUC is proposed by (Brzezinski & Stefanowski, 2017) to measure data stream classifiers' performance using the Area Under the ROC Curve (AUC) metric. To understand the approach, first, it is necessary to know its underlying concepts.

A predictor classifies an instance based on the probability of belonging to each of the classes (a score). To perform the classification, it has a decision threshold, that is, the minimum probability value needed to belong to a class. Varying the decision threshold allows us to obtain the classifier performance under different decision thresholds and plot the Receiver Operating Characteristics (ROC) curve. The ROC curve is a graph representing the relation between sensitivity and specificity of a classifier (each one is a graph axis), so a point on the curve refers to the classifier behavior with a particular operating condition (Brzezinski & Stefanowski, 2017; Flach, 2010).

AUC is a value that concentrates all the meaning of the ROC curve in a single point metric equivalent to the calculated area under the ROC curve (Sammut & Webb, 2010a). Its value ranges from 0 to 1, where a value closer to 1 is preferable.

According to the previous explanation, it is evident that it is necessary to perform several evaluation iterations, changing the decision threshold with the entire data to calculate the AUC. Therefore, this metric cannot be calculated as the data stream arrives. Consequently, (Brzezinski & Stefanowski, 2017) proposes a new metric named prequential AUC suitable for data stream scenarios.

The prequential AUC uses the red-black tree structure (Bayer, 1972; Guibas & Sedgewick, 1978) to sort the classification scores as a new sample arrives. Once the tree is updated, AUC is computed by the method proposed by (S. Wu et al., 2007). This method is equivalent to getting the sum of the

trapezoid's areas formed by each pair of consecutive points on the ROC curve (Brzezinski & Stefanowski, 2017).

MOA provides the prequential AUC calculation described in (Brzezinski & Stefanowski, 2017), through the BasicAUCImbalancedPerformanceEvaluator class inside its evaluation packet. Thus, our experiment uses this implementation. Also, the source code is available at <http://www.cs.put.poznan.pl/dbrzezinski/software.php>.

3.3 Experimental results and discussion

The prequential evaluation provides a set of values for each metric representing the fault diagnosis performance as the data arrive. Thus, each prequential metric is plotted in line-charts. This experiment yielded numerous results, exactly 250 performance graphs (five metrics for each of the twenty-five algorithms and both dataset files).

As Figure 5.13 shows, each graph represents the tested algorithms' performance curves according to the corresponding metric. The vertical axis indicates the metric values, and the horizontal axis indicates the number of incoming instances that have arrived until the measurement time. The black circles series are the results with the Diagnosis module approach, that is, with the rebalance stream strategy. The red squares series are the results of the online base algorithm performance without rebalancing or concept-drift treatment. Table 5.13 to Table 5.9 show examples of graphs obtained for all measured metrics and a few algorithms. All the 250 graphs are in Annex D.

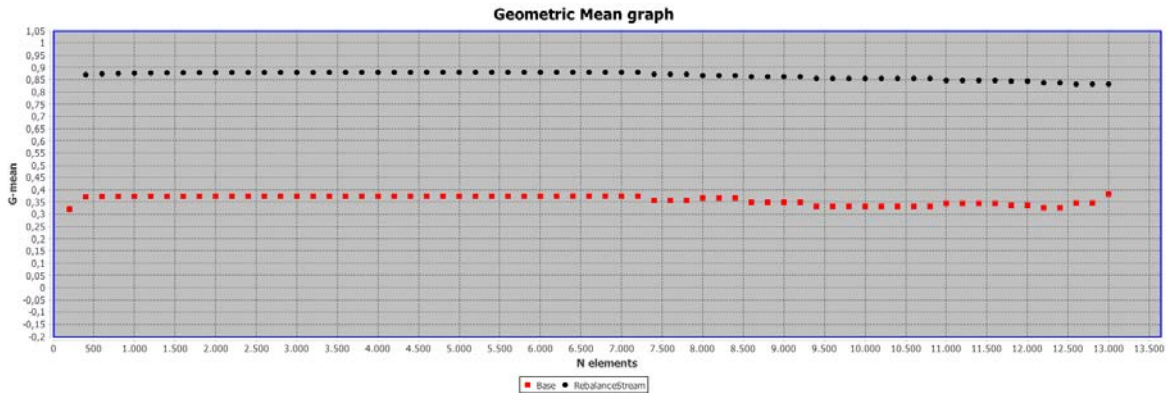


Figure 5.13 Performance curves format.

During the graphs reviewing, it was noted that there is a wide variety of behavior. The most common relationships that can be observed between the performance of the base algorithm and the rebalance stream approach are described below, and the indicated tables exemplify them.

- In most charts, the first two measurements are equivalent for both curves, and then the black curve is plotted higher than the red curve. All the plots of Table 5.9 to Table 5.12 depict that mentioned above.
- In some outcomes, both curves maintain an almost constant behavior throughout the measurement, so it is evident which one represents a better performance and if the difference between them is significant or not. Taking the examples from the next tables, this occurs in the second row of Table 5.10 and the first row of Table 5.11.

- A near-zero performance for base algorithm versus a better performance for rebalance stream strategy but also poor. The third row of Table 5.11 exemplifies it.
- Some results present a consistent and outstanding performance for rebalancing learning and a constant improvement throughout the base algorithm's performance measurement. The second row of Table 5.9 and the third row of Table 5.10 show it.
- For some graphs, there are sudden changes in either or both of the curves. From Table 5.9 to Table 5.12, there are examples of this behavior.
- There may be some constant changes in a measurement chunk, as observed in the first row and the first column of Table 5.10.
- Considerable decrease in the behavior of the base algorithm while the rebalancing algorithm improves. The second row of the Table 5.11 shows an example.
- The prequential AUC results are similar, so the black and red curves are close. This is evidenced by Table 5.13.

Table 5.9 Some results of prequential Sensitivity/Recall.

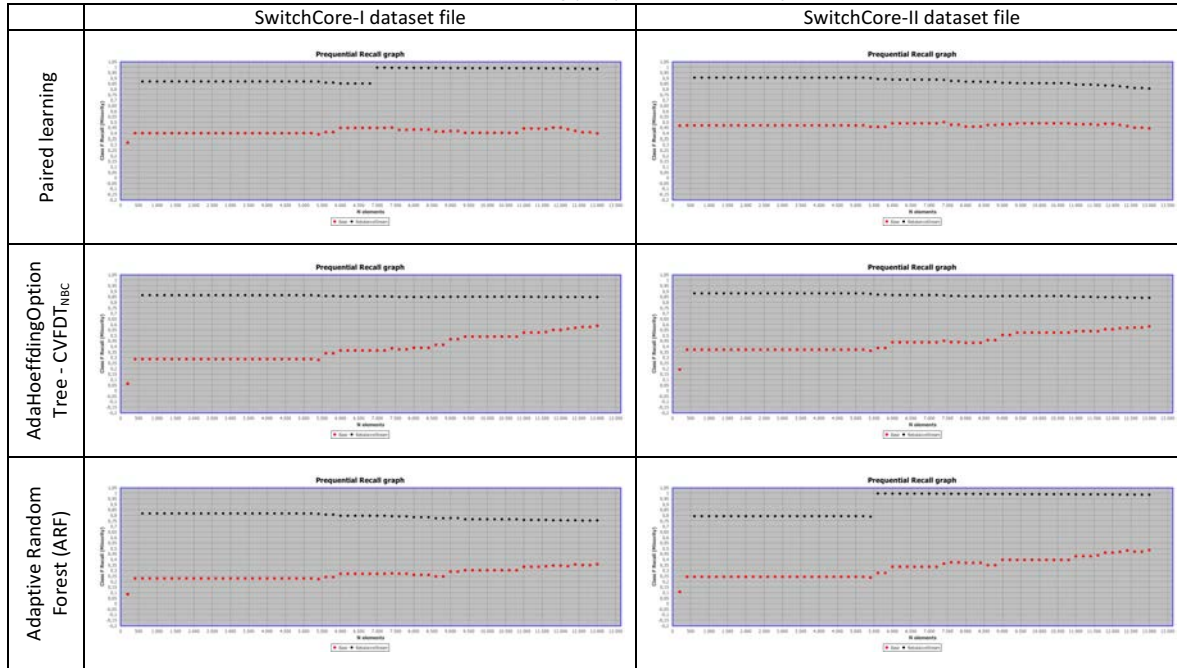
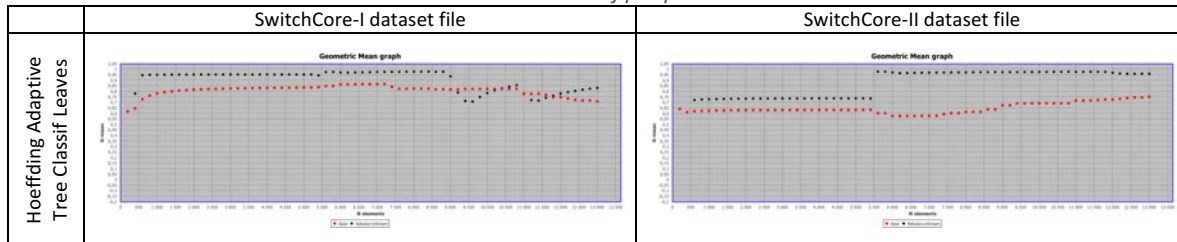


Table 5.10 Some results of prequential G-Mean.



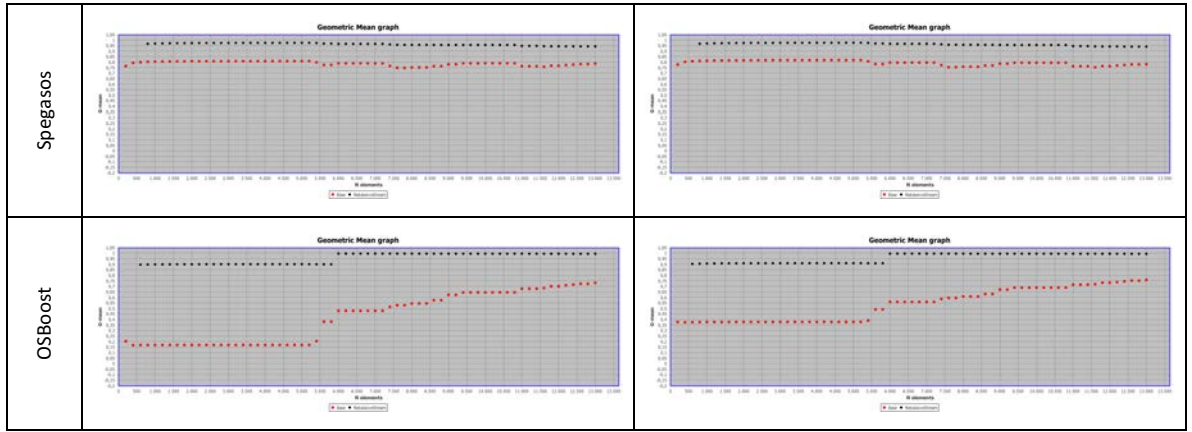


Table 5.11 Some results of prequential Kappa statistic.

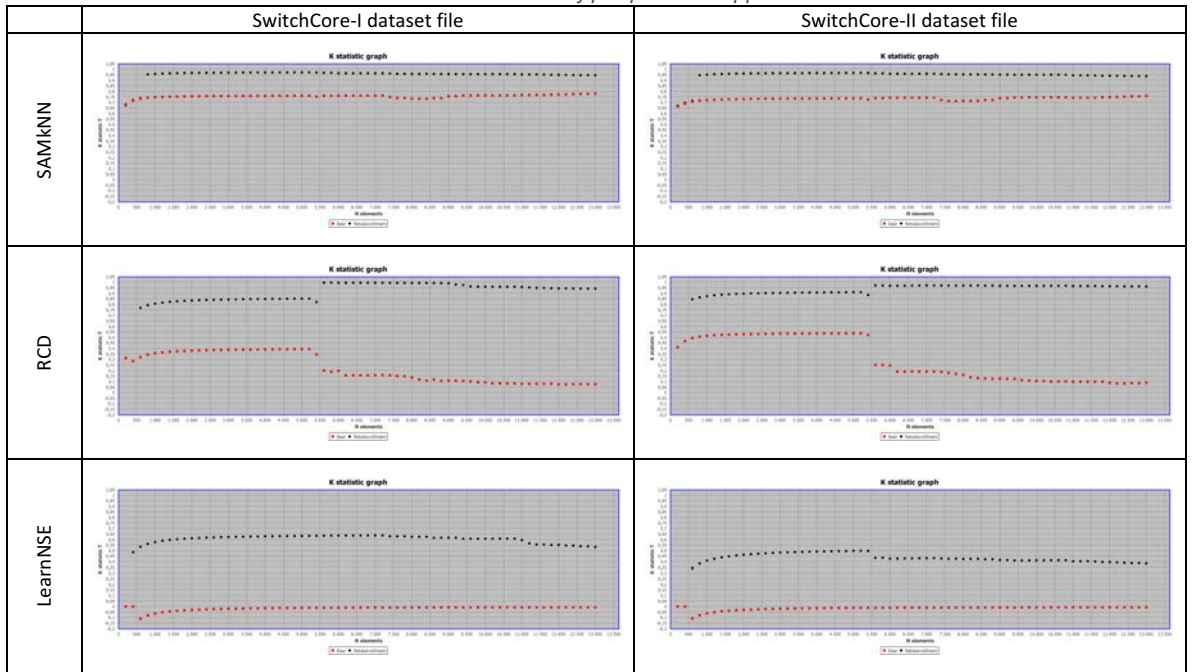
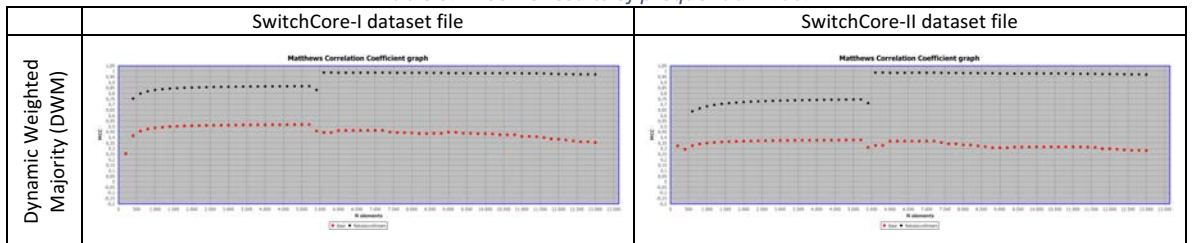


Table 5.12 Some results of prequential MCC.



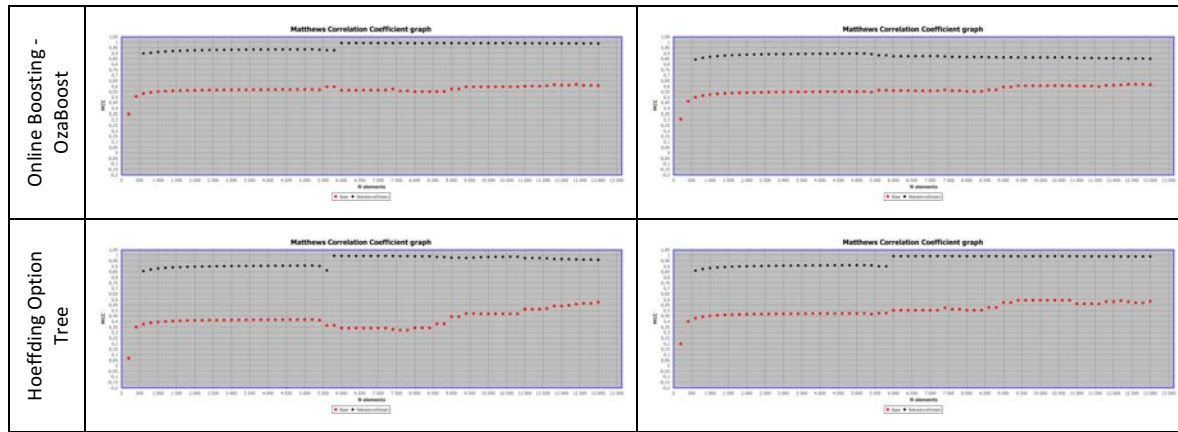
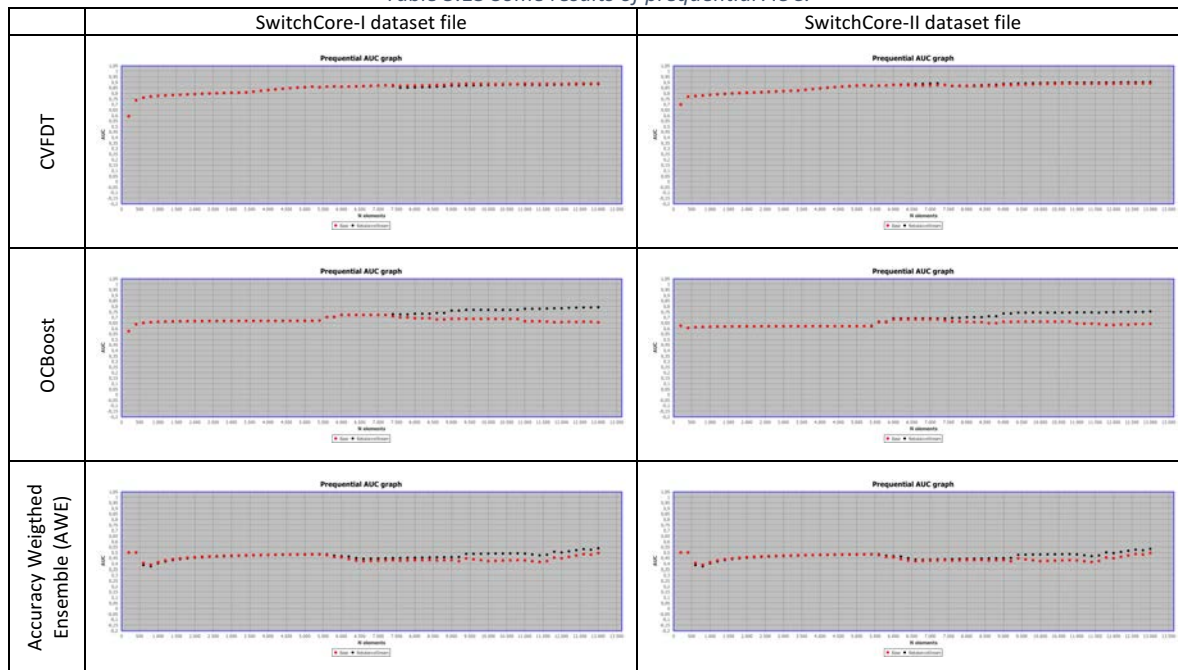


Table 5.13 Some results of prequential AUC.



The experiment yielded too many graphs to analyze, each one with different behaviors. Although the above chart samples are the overview of the graph types obtained, it is necessary to condense the results into tables to visually identify whether or not the rebalance strategy for fault classification represents an improvement of the base algorithm according to each performance metric. Hence, heatmaps were created for each of the metrics.

Three measurements were obtained to create the heatmaps and quickly identify the magnitude of the improvement represented by the use of the diagnosis module with the rebalancing approach. The calculation of these measures was also incorporated into the code of (Bernardo et al., 2019):

- ✓ the mean of the rebalanced curve (black curve),
- ✓ the mean of the curve without balancing (red curve),
- ✓ the weighted difference between them.

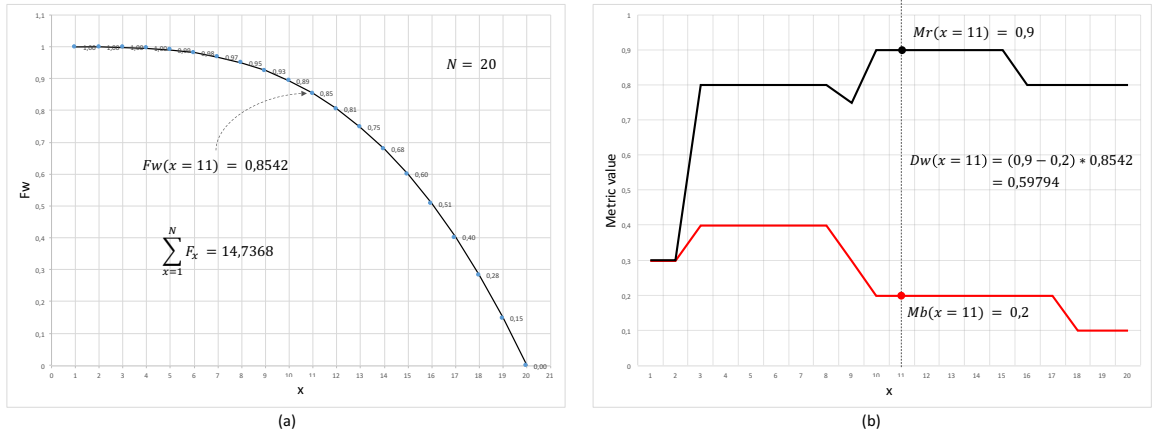
The weighted difference is a measure between 0 and 1 indicating the difference between the two performance lines (black and red line), taking into account a weighting function that assigns some importance to the performance obtained at each measurement.

Each example will become increasingly less significant to the overall average, as (Bifet et al., 2018) argues. Then, our weighting function is a decreasing function that assigns a greater value to the differences of the first measurements. Equation (36) represents the weighting function (Fw). N is the total of measurements. x indicates the measurement number ($x = 1$ for the first measurement, $x = 2$ for the second, and so on). The k value is set to 3 to obtain a convex exponential weight that maintains a high valuation of differences in an initial interval and then decreases. As an example, Figure 5.14(a) shows the weighting function for a prequential evaluation of 20 measurements.

Equation (37) represents the weighted difference (Dw). Mr_x is the metric value obtained at point x with the rebalancing strategy (point of black curve). Mb_x is the metric value obtained at point x with the base algorithm (point of red curve). As the equation shows, it is the summation of the weighted differences at each point, and the result is normalized to obtain a value between 0 and 1. Figure 5.14 illustrates the equation components.

$$Fw(x) = 1 - \left(\frac{x-1}{N-1} \right)^k ; \quad 1 \leq x \leq N , \quad k = 3 \quad (36)$$

$$Dw = \frac{\sum_{x=1}^N (Mr_x - Mb_x) F_x}{\sum_{x=1}^N F_x} \quad (37)$$



(a) Example of the weighting function for a prequential evaluation of 20 measurements

(b) Example of how to compute the weighted difference of a point

Figure 5.14 Weighted difference elements.

For each performance metric (sensitivity, G-mean, kappa, MCC, and prequential AUC), three heatmaps were obtained representing the values of the three measures mentioned above, for the 25 algorithms in Table 5.8. Figure 5.15 contains the color settings used in the heatmaps. Figure 5.15(a) shows the configuration for the means. The redder the color, the better the measurement because it is closer to 1, while, if the color becomes lighter, the value is closer to 0, and worse is the performance. Figure 5.15(b) indicates the color setting for the weighted difference. A positive difference is colored green, and the more saturated the color, the more significant the improvement. The gray color indicates no difference and the blue color that there is a negative difference. The difference also is reflected if one of the means heatmaps is lighter than the other.

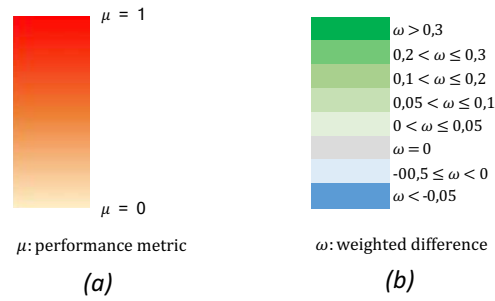


Figure 5.15 The color scale for evaluation metrics.

Figure 5.16 to Figure 5.20 present the created heatmaps. As can be seen, some image fields are set as "NaN," which means that it was impossible to calculate the metric at some points (due to very bad behavior), making the developed tool unable to calculate the corresponding metric's means.

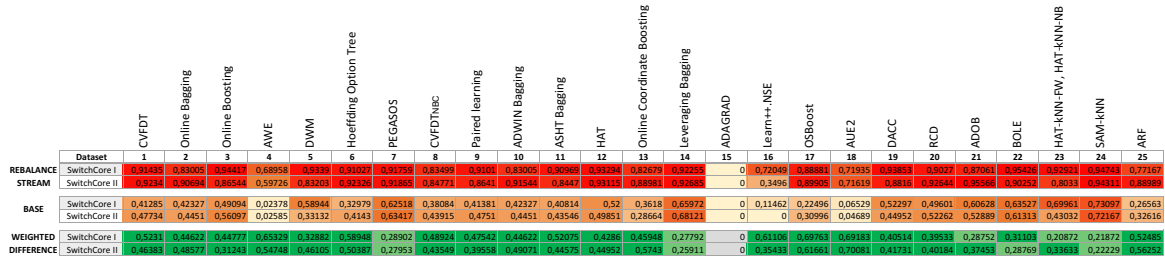


Figure 5.16 Heatmap of prequential Sensitivity/Recall results.

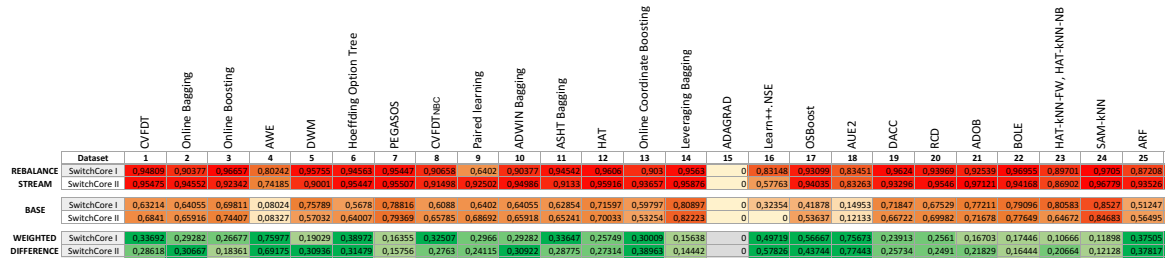


Figure 5.17 Heatmap of prequential G-mean results.

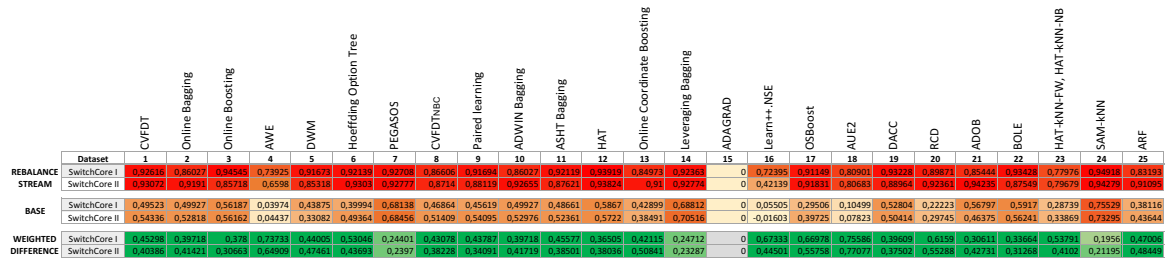


Figure 5.18 Heatmap of prequential Kappa statistic results.

	Dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		CVFDT	Online Bagging	Online Boosting	AWE	DWM	Hoeffding Option Tree	PEGASOS	CVFDTmc	Paired learning	ADWIN Bagging	ASHT Bagging	HAT	Online Coordinate Boosting	Leveraging Bagging	ADAGRAD	Learn++-NSE	OSBoost	AUE2	DACC	RCD	ADOB	BOLE	HAT-kNN-FW, HAT-kNN-NB	SAM-kNN	ARF
REBALANCE	SwitchCore I	0.92774	0.86193	0.94613	NaN	0.81675	0.92295	0.92745	0.86818	0.51783	0.86193	0.92237	0.94635	0.85105	0.9239	NaN	NaN	0.91409	NaN	0.93335	0.89998	0.85448	0.9344	0.79648	0.94929	0.83847
	SwitchCore II	0.9318	0.92077	0.85722	NaN	0.85963	0.93139	0.92611	0.8729	0.88215	0.92822	0.87852	0.93964	0.91303	0.92794	NaN	NaN	0.92147	NaN	0.85014	0.92428	0.94236	0.87566	0.80855	0.94286	0.91437
BASE	SwitchCore I	0.51353	0.51389	0.57057	NaN	0.45367	0.41708	0.68545	0.48931	0.46219	0.51389	0.50255	0.59561	0.44162	0.68961	NaN	NaN	0.32757	NaN	0.52857	0.2591	0.56846	0.59257	0.3495	0.76025	0.43408
	SwitchCore II	0.55295	0.54461	0.56199	NaN	0.33862	0.507	0.68786	0.52577	0.54917	0.54653	0.54212	0.58354	0.41863	0.70621	NaN	NaN	0.43096	NaN	0.51033	0.33885	0.46806	0.56357	0.35436	0.74802	0.47317
WEIGHTED	SwitchCore I	0.43304	0.38188	0.3693	NaN	0.42818	0.51208	0.24074	0.49729	0.43185	0.38186	0.43853	0.35573	0.41243	0.24564	NaN	NaN	0.43719	NaN	0.29967	0.58882	0.30568	0.33573	0.446374	0.19063	0.42303
	SwitchCore II	0.39377	0.39669	0.39646	NaN	0.47387	0.42454	0.23715	0.37093	0.31248	0.39917	0.36571	0.36842	0.4803	0.23185	NaN	NaN	0.52185	NaN	0.36995	0.51998	0.42276	0.31155	0.41152	0.20462	0.44963

Figure 5.19 Heatmap of prequential MCC results.

	Dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		CVFDT	Online Bagging	Online Boosting	AWE	DWM	Hoeffding Option Tree	PEGASOS	CVFDTmc	Paired learning	ADWIN Bagging	ASHT Bagging	HAT	Online Coordinate Boosting	Leveraging Bagging	ADAGRAD	Learn++-NSE	OSBoost	AUE2	DACC	RCD	ADOB	BOLE	HAT-kNN-FW, HAT-kNN-NB	SAM-kNN	ARF
REBALANCE	SwitchCore I	0.84002	0.91582	0.93697	0.47046	0.800799	0.82507	0.81083	0.84033	0.88174	0.81582	0.91879	0.85591	0.71464	0.92124	0.5	0.52552	0.90388	0.45182	0.87877	0.8298	0.91286	0.92452	0.87253	0.88742	0.85588
	SwitchCore II	0.85766	0.92718	0.93561	0.46638	0.67701	0.8463	0.81355	0.85167	0.91042	0.92862	0.92286	0.86784	0.67805	0.93464	0.5	0.54407	0.90573	0.45443	0.90253	0.87655	0.93143	0.94626	0.87663	0.88567	0.93942
BASE	SwitchCore I	0.84528	0.91525	0.89356	0.44997	0.83787	0.83822	0.80975	0.84341	0.88876	0.9195	0.92112	0.86124	0.5765	0.92166	0.5	0.49816	0.89934	0.45322	0.87862	0.85487	0.89615	0.92882	0.89658	0.88488	0.94236
	SwitchCore II	0.85326	0.92571	0.93422	0.44991	0.68197	0.83754	0.81419	0.85224	0.91128	0.9258	0.9254	0.86923	0.64072	0.93875	0.5	0.49829	0.90185	0.45558	0.90001	0.8819	0.81159	0.94629	0.89258	0.88238	0.94684
WEIGHTED	SwitchCore I	-0.00401	5.03E-04	0.03961	0.01198	-0.02368	-0.00905	6.25E-04	-0.00237	-3.22E-04	2.34E-04	-0.00218	-0.012	0.0198	-1.12E-04	0	0.03011	0.003	0.01693	-1.79E-04	-0.01743	0.02103	-0.00285	-0.01436	0.00221	-0.00741
	SwitchCore II	0.00256	0.00113	7.70E-04	0.00884	-0.00452	0.00622	-3.08E-04	-0.00117	-0.00104	0.00167	-0.00153	-0.00134	0.0211	-0.00154	0	0.04419	0.00337	-0.00998	0.00129	-0.00289	0.07732	0.00639	-0.02765	0.0029	-0.00773

Figure 5.20 Heatmap of prequential AUC results.

According to Figure 5.16, the sensitivity during fault diagnosis is higher when using the rebalancing and concept-drift approach than when using only the base incremental learning algorithm. So, the PALADIN diagnosis module learns from minority class despite huge class imbalance. This behavior is attributed to the fact that SMOTE was able to engage in online learning, positively affecting failure classification.

The prequential measurements of G-Mean (Figure 5.17), kappa (Figure 5.18), and MCC (Figure 5.19) also confirm that the Diagnosis module does not neglect the learning of any of the two states of the network (in failure and healthy states).

The intensity of the color of the weighted difference heatmap for the Sensitivity, G-Mean, kappa, and MCC metrics also suggests that it is not enough to use an incremental algorithm without the components of the proposed Diagnosis module (concept drift detector and class imbalance detector). In the same line, it is not enough to deal with the concept-drift to face the classification of an imbalanced data stream. As evidence, most of the base algorithms have concept drift incorporated (as Table 5.8 indicates in experiment setup); however, they perform poorly, so a rebalancing and concept-drift detection procedure external to the algorithm is necessary.

Meanwhile, prequential AUC is very similar in the proposed Diagnosis module and the online base classifier (Figure 5.20). Suppose this is contrasted with the results mentioned above. In that case, it is safe to say that this metric does not provide reliable information to compare the two scenarios and neither to evaluate the classification performance of imbalanced data streams.

If it were to select an online base algorithm for the Diagnosis module, it is evident that some do not perform well even with the rebalancing process and could not be selected as candidates to be implemented within the Diagnostic module (AWE, ADAGRAD, Learn++-NSE, and AUE2). However, this occurs with few. Twenty-one algorithms out of twenty-five have excellent results if they are coupled with the concept-drift and class imbalance detectors, as the diagnosis module proposes.

Hence, this proposal is a useful approach to network fault diagnosis and appropriate for network scenarios where monitoring data arrives on the fly.

This experiment reaffirms that the PALADIN model proposed to detect failures from peripheral devices' monitoring data allows detecting internal network failures using data stream learning techniques.

Chapter VI: Conclusion and future work

This research aimed to diagnose IP-based networks' faults in a timely, non-invasive way and resilient to dynamic network changes. The proposed PALADIN model (Peripheral Fault Diagnosis Model for IP-based Networks) contains the necessary elements for the continuous and timely diagnosis of IP-based network faults; it introduces the idea of periodical monitorization of peripheral network elements and uses data analytics techniques to process it. Based on the analysis, processing, and classification of peripherally collected data, it can be concluded that PALADIN achieves the objective. The results indicate that the peripheral monitorization allows diagnosing faults in the internal network; besides, the diagnosis process needs an incremental learning process, concept-drift detection elements, and rebalancing strategy.

This research's point of departure was the hypothesis of a fault propagation phenomenon that allows the observation of failure symptoms at a higher network level than the fault origin. Thus, for the model's construction, monitoring data was collected from an extensive campus network in which impact link failures were induced at different instants of time and with different duration. These data correspond to widely used parameters in the actual management of a network. The collected data allowed us to understand the faults' behavior and how they are manifested at a peripheral level.

Based on this understanding and a data analytics process, the first three modules of our model were proposed (Identify, Collection and Structuring), which define the data collection peripherally and the necessary data pre-processing to obtain the description of the network's state at a given moment. These modules give the model the ability to structure the data taking into account the delays of the multiple responses that the network delivers to a single monitoring probe and the multiple network interfaces that a peripheral device may have.

Thus, a structured data stream is obtained, and it is ready to be analyzed. For this analysis, it was necessary to implement an incremental learning framework that respects networks' dynamic nature. It comprises three elements, an incremental learning algorithm, a data rebalancing strategy, and a concept drift detector. This framework is the Diagnosis module of the PALADIN model.

The results of the experiments were as expected. On the one hand, impact failures propagate in the network and manifest themselves in the monitored network's peripheral devices. On the other, PALADIN makes it possible to learn from these network manifestations and diagnose internal network failures. The latter was verified with 25 different incremental algorithms, ADWIN as concept-drift detector and SMOTE (adapted to streaming scenario) as the rebalancing strategy.

This research clearly illustrates that it is unnecessary to monitor all the internal network elements to detect a network's failures. It is enough to choose the peripheral elements to be monitored. This way, the increase in network traffic and control overhead is avoided. Furthermore, with proper processing of the collected status and traffic descriptors, it is possible to learn from the arriving data using incremental learning in cooperation with data rebalancing and concept drift approaches. This proposal continuously diagnoses the network symptoms without leaving the system vulnerable to failures while being resilient to the network's dynamic changes.

Having solved the fault detection problem, then the question of how granular the peripheral approach can detect faults arises; for example, determine if a failure occurs in the access layer or the distribution layer, and even determine what causes failure.

Future studies could address the granularity concerns mentioned above from the peripheral perspective to understand our results' implications better. Also, we would like to extend our study to overlapping faults because it is an interesting point that has not been answered with the PALADIN model. Another important aspect not specific to networking is the normalization process for a data stream. In the experimentation, this process was carried out before the incremental learning process, as if all the data arrived normalized. However, the calculation of statistical measures from a non-static data set is still a field under study.

On the other hand, researchers should consider that fault detection handled as a classification process is a problem with combined difficulties from streaming and excessive imbalanced data. This is why careful metrics selection is needed for performance evaluation, and as a recommendation, the sequential AUC is not a suitable performance metric for unbalanced data streams.

To summarize this research's main contributions, let us return to the problem statement identified from a strong literature review. It ensures that there is no non-invasive technique to continuously diagnose the network symptoms without leaving the system vulnerable to any failures, nor a resilient technique to the network's dynamic changes, which can cause new failures with different symptoms. This research first evidences the phenomenon of impact fault propagation, making it possible to detect fault symptoms at a monitored network's peripheral level. It translates into non-invasive monitoring of the network. Second, the PALADIN model is the major contribution in the fault detection context because it covers two aspects. An online learning model to continuously process the network symptoms and detect internal failures. Moreover, the concept-drift detection and rebalance data stream components which make resilience to dynamic network changes possible. Third, it is well known that the number of available real-world datasets for testing imbalanced stream classifiers is still too small. That number is further reduced for networking context. The SOFI dataset obtained with the first modules of the PALADIN model contributes to that number and encourages works related to unbalanced data streams and those related to network fault diagnosis.

As previously noted, the main objective of this research was achieved because PALADIN is a model for the continuous and timely diagnosis of IP-based networks faults, independent of the network structure and based on data analytics techniques. Furthermore, the three specific objectives pursued were achieved as follows:

1. A data set that relates peripheral symptoms with network faults was created and named SOFI.

2. PALADIN's diagnosis module represents an online fault classification model with data stream considerations.
3. The implementation and evaluation of PALADIN through the experiments allowed us to measure the proposed model behavior over time.

Annexes

Annex A: Network Emulation on GNS3

This network configuration is based on the "Enterprise Network on GNS3" online guide published by (Brezula, 2017). Figure A.1 shows the emulated network over the GNS3 platform; it details all the addressing and routing configuration, and all the device details (vendor, software version). Each item details the configuration of each device in the campus infrastructure layers, the server farm, the enterprise edge, and the service provider edge. Finally, summary credentials are presented.

A.1 Access Layer Configuration

A.1.1 Access Switches Configuration

The two access switches are similarly configured. Therefore, the configuration for one Access Switch (AccSwitch-I) is indicated, and both configuration files are attached to perform the same procedure with the remaining access switch (AccSwitch-II). Both switches are emulated by installed qcow2 images of Cisco vIOS-l2 Qemu appliance (version 15.2).

a) Basic Configuration

```
switch# conf t
switch(config)# hostname AccSwitch-I
```

b) Configuration of VLANs

AccSwitch-I provides the connection to VLAN10 and VLAN20, while AccSwitch-II provides it to VLAN30 and VLAN40.

```
AccSwitch-I(config)# vlan 10
AccSwitch-I(config-vlan)# no shutdown
AccSwitch-I(config)# vlan 20
AccSwitch-I(config-vlan)# no shutdown
AccSwitch-I(config-vlan)# exit
```

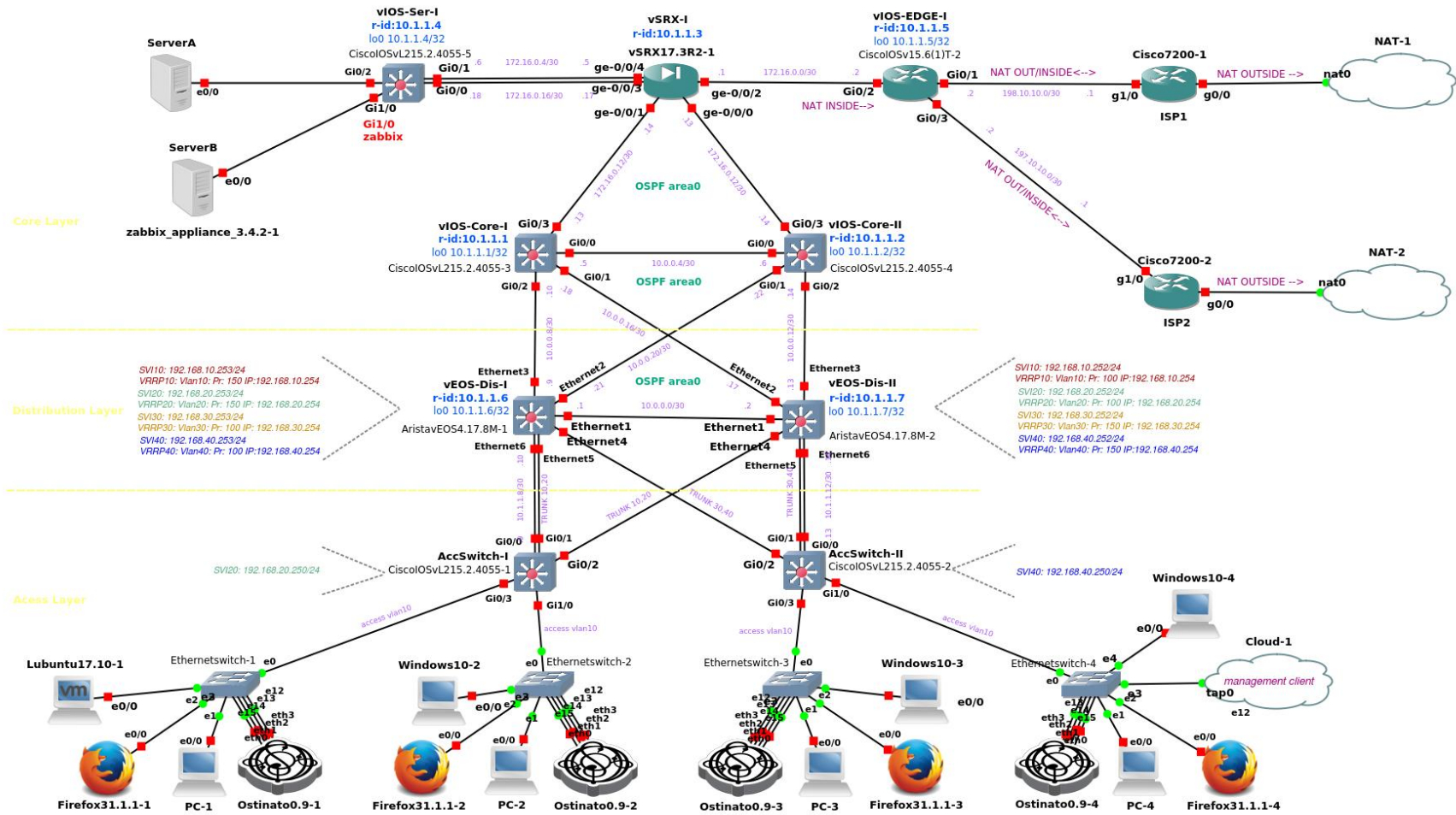


Figure A.1 Emulated network in GNS3 platform.

c) IP Address and Trunk Port Configuration

The port G0/0 is selected as the management port. This interface has an IP address and netmask appropriate to access it remotely. Therefore, it is configured as a routed port or a Layer 3 interface. 10.1.1.8/30 is the subnetwork to which it belongs and shared with DisSwitch-I of the distribution layer.

```
AccSwitch-I(config)# interface GigabitEthernet 0/0
AccSwitch-I(config-if)# description Link to DisSwitch-I
AccSwitch-I(config-if)# no switchport
AccSwitch-I(config-if)# ip static 10.1.1.9 255.255.255.252
AccSwitch-I(config-if)# no shutdown
AccSwitch-I(config-if)# exit
```

In order to allow these switches to access servers in the data center (to synchronize their time with the NTP server, to send logs to the Rsyslog server, and so forth), the Switch Virtual Interface (SVI) is created in the access switches.

```
AccSwitch-I(config)# interface vlan 20
AccSwitch-I(config-if-vlan)# ip address 192.168.20.250 255.255.255.0
AccSwitch-I(config-if-vlan)# no shutdown
AccSwitch-I(config-if-vlan)# exit
```

The ports GigabitEthernet0/1 and 0/2 on access switches are trunk ports because they carry traffic from multiple VLANs. Also, they connect the access switches to the distribution network elements. The ports which connect access switches with the ethernet switches (on the same access layer) are GigabitEthernet0/3 and 1/0, and they are configured as access ports.

```
AccSwitch-I(config)# interface GigabitEthernet0/1
AccSwitch-I(config-if)# switchport trunk encapsulation dot1q
AccSwitch-I(config-if)# switchport mode trunk
AccSwitch-I(config-if)# switchport trunk allowed vlan 10,20
AccSwitch-I(config-if)# no shutdown
AccSwitch-I(config-if)# exit
```

```
AccSwitch-I(config-if)# interface GigabitEthernet0/2
AccSwitch-I(config-if)# switchport trunk encapsulation dot1q
AccSwitch-I(config-if)# switchport mode trunk
AccSwitch-I(config-if)# switchport trunk allowed vlan 10,20
AccSwitch-I(config-if)# no shutdown
AccSwitch-I(config-if)# exit
```

```
AccSwitch-I(config-if)# interface GigabitEthernet0/3
AccSwitch-I(config-if)# switchport mode access
AccSwitch-I(config-if)# switchport access vlan 10
AccSwitch-I(config-if)# no shutdown
```

```
AccSwitch-I(config-if)# int GigabitEthernet1/0
AccSwitch-I(config-if)# switchport mode access
AccSwitch-I(config-if)# switchport access vlan 20
AccSwitch-I(config-if)# no shutdown
```


Creating a static default route for the switch for routing NTP and Syslog messages to the server farm is necessary.

```
AccSwitch-I(config)# ip route 0.0.0.0 0.0.0.0 192.168.20.254
```

d) NTP Configuration

The NTP server is located in 172.16.50.1. The timezone is UTC-5 for Colombia.

```
AccSwitch-I(config)# ntp server 172.16.50.1
AccSwitch-I(config)# ntp source GigabitEthernet 0/0
AccSwitch-I(config)# clock timezone UTC-5 -5 0
```

e) Logging

The level of logging corresponds to notifications (level 5).

```
AccSwitch-I(config)# logging trap notifications
```

Configure the Syslog server address where the logs are sent (Server B with IP address 172.16.50.3).

```
AccSwitch-I(config)# logging host 172.16.50.3
```

The logging source interface is the management interface or loopback.

```
AccSwitch-I(config)# logging source-interface GigabitEthernet0/0
```

f) SNMP Configuration

Enable the Read-only (RO) community string as "emulation" in this particular case.

```
AccSwitch-I(config)# snmp-server community emulation RO
```

Enable the router to send all traps to the Zabbix Server (172.16.50.3) with the community string "emulation" and SNMP version 2.

```
AccSwitch-I(config)# snmp-server host 172.16.50.3 traps version 2c emulation
AccSwitch-I(config)# snmp-server enable traps
```

g) Configuration Files

The `/enterprise_campus/access/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration files for the AccSwitch-I and AccSwitch-II devices for consultation purposes.

A.1.2 Clients Configuration

Clients are default DHCP configured (172.16.50.1 as DHCP server). However, VLAN 40 is used for network management tasks, so all its devices are statically configured. The client device types used are Basic PC (GNS3 built-in basic Linux PC), Ostinato appliance, Firefox appliance, Windows 10 appliance, PC Loopback (connection to the machine itself), and Lubuntu virtual machine.

The static IP address configuration for Basic PC and PC Loopback is in the next item. The other devices are configured with an intuitive graphical interface, so they will not be explained.

a) Network Configuration for Basic PC

The bootlocal.sh file contains network interface configuration information for Basic PC. Access this file in PC4.

```
$ vi /opt/bootlocal.sh
```

To setup eth0 to a static address, enter:

```
hostname PC4
ifconfig eth0 192.168.40.1 netmask 255.255.255.0
route add default gw 192.168.40.254
echo "nameserver 172.16.50.1" > /etc/resolv.conf
```

The next command is needed to save the configuration.

```
$ /usr/bin/filetool.sh -b
```

b) Loopback Configuration

A loopback interface is configured to use the GNS3 user interface machine as an emulated network client. For this purpose, follow the next steps in that machine.

```
$ sudo apt-get install uml-utilities
$ sudo modprobe tun
```

Verify the name of the loopback interface available for configuration.

```
$ sudo tuncctl
Set 'tap0' persistent and owned by uid 0
```

Configure the available loopback interface with an IP within the VLAN 40 addresses range.

```
$ sudo ifconfig tap0 192.168.40.2 netmask 255.255.255.0
$ sudo route add default gw 192.168.40.254
$ sudo echo "nameserver 172.16.50.1" >> /etc/resolv.conf
```

A Cloud element has to be added and configured to add the loopback interface as a client of the emulated network, as shown in Figure A.2:

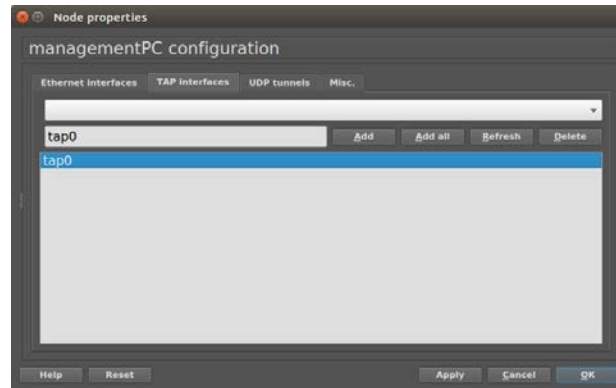


Figure A.2 Configuration of Cloud element with the loopback interface.

A.2 Distribution Layer Configuration

The two distribution switches are similarly configured. Therefore, the configuration for one Distribution Switch (DisSwitch-I) is indicated, and both configuration files are attached to perform the same procedure with the remaining distribution switch (DisSwitch-II). Both switches are Arista vEOS Qemu appliances installed on VMware disks (version 4.17.8M).

a) Basic Configuration

To configure the Arista network elements, it is necessary to log into the appliance using the default username **admin**, which has no assigned password. Then, configure the hostname as the first step.

```
switch> en
switch# conf t
switch(config)# hostname DisSwitch-I
```

b) Configuration of VLAN

```
DisSwitch-I(config)# vlan 10
DisSwitch-I(config-vlan-10)# vlan 20
DisSwitch-I(config-vlan-20)# vlan 30
DisSwitch-I(config-vlan-30)# vlan 40
DisSwitch-I(config-vlan-40)# exit
```

c) IP Address and Trunk Port Configuration

Use the loopback interface for management and assign the IP address 10.1.1.6/32 to it.

```
DisSwitch-I(config)# interface loopback 0
DisSwitch-I(config-if-Lo0)# ip address 10.1.1.6/32
```

On both distribution switches, the ports Ethernet4 and Ethernet5 are configured as trunks because they are Layer 2 interfaces that carry traffic from VLANs.

```
DisSwitch-I(config)# interface Ethernet5
DisSwitch-I(config-if-Et4)# description Link to AccSwitch-I
DisSwitch-I(config-if-Et4)# switchport
DisSwitch-I(config-if-Et4)# switchport mode trunk
DisSwitch-I(config-if-Et4)# switchport trunk allowed vlan 10,20
DisSwitch-I(config-if-Et4)# no shutdown
DisSwitch-I(config-if-Et4)# exit

DisSwitch-I(config)# interface Ethernet4
DisSwitch-I(config-if-Et5)# description Link to AccSwitch-II
DisSwitch-I(config-if-Et5)# switchport
DisSwitch-I(config-if-Et5)# switchport mode trunk
DisSwitch-I(config-if-Et5)# switchport trunk allowed vlan 30,40
DisSwitch-I(config-if-Et5)# no shutdown
DisSwitch-I(config-if-Et4)# exit
```

On both distribution network elements, the port Ethernet6 is a routing interface that connects the management port of the access switches to the network. As is clear, the loopback and Ethernet 6 ports belong to the same network 10.1.1.0/24.

```
DisSwitch-I(config)# interface Ethernet6
DisSwitch-I(config-if-Et6)# description Link to Management AccSwitch-I
DisSwitch-I(config-if-Et6)# no switchport
DisSwitch-I(config-if-Et6)# ip address 10.1.1.10/30
DisSwitch-I(config-if-Et6)# no shutdown
DisSwitch-I(config-if-Et6)# exit
```

The routed ports or Layer 3 interfaces (Ethernet 1, 2, and 3) connect the distribution network elements to each other and to the Core switches. The networks formed are shown in Table A.1. As is clear, all point-to-point links belong to the subnet 10.0.0.0/24.

Table A.1 Distribution networks.

Network	Interfaces
10.0.0.0/30	Ethernet 1 of DisSwitch-I and Ethernet 1 of DisSwitch-II
10.0.0.2/30	Ethernet 2 of DisSwitch-I and Ethernet 1 of CoreSwitch-II
10.0.0.8/30	Ethernet 3 of DisSwitch-I and Ethernet 2 of CoreSwitch-I
10.0.0.16/30	Ethernet 2 of DisSwitch-II and Ethernet 1 of CoreSwitch-I
10.0.0.12/30	Ethernet 3 of DisSwitch-II and Ethernet 2 of CoreSwitch-II

```
DisSwitch-I(config)# interface Ethernet1
DisSwitch-I(config-if-Et3)# description Link to DisSwitch-II
DisSwitch-I(config-if-Et3)# no switchport
DisSwitch-I(config-if-Et3)# ip address 10.0.0.1/30
DisSwitch-I(config-if-Et3)# no shutdown
DisSwitch-I(config-if-Et3)# exit
```

```
DisSwitch-I(config)# interface Ethernet2
DisSwitch-I(config-if-Et1)# description Link to CoreSwitch-II
DisSwitch-I(config-if-Et1)# no switchport
```

```

DisSwitch-I(config-if-Et1)# ip address 10.0.0.21/30
DisSwitch-I(config-if-Et1)# no shutdown
DisSwitch-I(config-if-Et1)# exit

DisSwitch-I(config)# interface Ethernet3
DisSwitch-I(config-if-Et2)# description Link to CoreSwitch-I
DisSwitch-I(config-if-Et2)# no switchport
DisSwitch-I(config-if-Et2)# ip address 10.0.0.9/30
DisSwitch-I(config-if-Et2)# no shutdown
DisSwitch-I(config-if-Et2)# exit

```

To prevent connecting with another network device accidentally, shutdown the unused interfaces.

```

DisSwitch-I(config)# interface Ethernet7
DisSwitch-I(config-if-Et7)# description Unused
DisSwitch-I(config-if-Et7)# shutdown
DisSwitch-I(config-if-Et7)# exit

```

d) Switch Virtual Interfaces Configuration

In order to route traffic between VLANs, the configuration of Switch Virtual Interfaces (SVI) is necessary for each VLAN. The interface SVI10, 20, 30, and 40 of DisSwitch-I has the IP address 192.168.x.253/24 configured, where x is the VLAN identifier. Those SVI interfaces have the IP address 192.168.x.252/24 assigned on the switch DisSwitch-II.

The following commands execute such configuration. That configuration snippet must be repeated for each VLAN changing the x for the corresponding VLAN identifier (10, 20, 30, or 40).

```

DisSwitch-I(config)# interface vlan x
DisSwitch-I(config-if-V110)# ip address 192.168.x.253/24
DisSwitch-I(config-if-V110)# no shutdown
DisSwitch-I(config-if-V110)# exit

```

e) OSPF Protocol Configuration

Enable IP routing on the distribution network elements.

```

DisSwitch-I(config)# ip routing

```

Open Shortest Path First (OSPF) configuration ensures the routes propagation inside the campus and server farm. In order to avoid adjacency and save CPU cycles of the distribution switches, the interfaces Ethernet4, Ethernet5, and Ethernet6 are configured as passive. In this way, no routing updates are performed on the SVI interfaces, the trunk ports, and the connection to the management interface of the access switches.

```

DisSwitch-I(config)# router ospf 1
DisSwitch-I(config-router-ospf)# router-id 10.1.1.6
DisSwitch-I(config-router-ospf)# network 10.1.1.6/32 area 0
DisSwitch-I(config-router-ospf)# network 10.1.1.8/30 area 0
DisSwitch-I(config-router-ospf)# network 10.0.0.0/30 area 0
DisSwitch-I(config-router-ospf)# network 10.0.0.20/30 area 0
DisSwitch-I(config-router-ospf)# network 10.0.0.8/30 area 0

```

```

DisSwitch-I(config-router-ospf)# network 192.168.10.0/24 area 0
DisSwitch-I(config-router-ospf)# network 192.168.20.0/24 area 0
DisSwitch-I(config-router-ospf)# network 192.168.30.0/24 area 0
DisSwitch-I(config-router-ospf)# network 192.168.40.0/24 area 0
DisSwitch-I(config-router-ospf)# passive-interface Ethernet4
DisSwitch-I(config-router-ospf)# passive-interface Ethernet5
DisSwitch-I(config-router-ospf)# passive-interface Ethernet 6
DisSwitch-I(config-router-ospf)# passive-interface vlan 10,20,30,40

```

In order to securely exchange routing updates, the Message-Digest algorithm 5 (MD5) is used to set the password authentication for OSPF neighbors. The OSPF broadcast network type used is Point-to-Point.

```

DisSwitch-I(config)# interface Ethernet2
DisSwitch-I(config-if-Et1)# ip ospf authentication message-digest
DisSwitch-I(config-if-Et1)# ip ospf message-digest-key 1 md5 #MyPass!034
DisSwitch-I(config-if-Et1)# ip ospf network point-to-point

DisSwitch-I(config-if-Et1)# int Ethernet3
DisSwitch-I(config-if-Et2)# ip ospf authentication message-digest
DisSwitch-I(config-if-Et2)# ip ospf message-digest-key 1 md5 #MyPass!034
DisSwitch-I(config-if-Et2)# ip ospf network point-to-point

DisSwitch-I(config-if-Et2)# int Ethernet1
DisSwitch-I(config-if-Et3)# ip ospf authentication message-digest
DisSwitch-I(config-if-Et3)# ip ospf message-digest-key 1 md5 #MyPass!034
DisSwitch-I(config-if-Et3)# ip ospf network point-to-point

```

f) Virtual Router Redundancy Protocol (VRRP) Configuration

Each VLAN has a default gateway as a single connection point to the campus network. This point is a distribution switch; then, if this network element fails, the connected VLANs will be decoupled from the network. It is necessary to configure all the gateways IP addresses in both distribution switches to avoid that. Thanks to the VRRP protocol, the above is possible, enabling several network elements to use the same virtual IP address. So, one distribution switch is configured as the Master while the other acts as a backup when the Master becomes unavailable.

The switch DisSwitch-I is the Master for the VLAN10 and 20, so it has the IP addresses 192.168.10.254 and 192.168.20.254 assigned (default gateway). DisSwitch-I is the VRRP Backup for the VLAN30 and 40, so it forwards traffic from these VLANs in case the Master (DisSwitch-II) becomes unavailable. In the same way, the DisSwitch-II is the Master for VLAN30 and 40 and the Backup for VLAN10 and 20.

In order to determine whether the switch becomes a Master, the higher VRRP priority is configured for the wanted SVI interface. For example, DisSwitch-I sets 150 as VRRP priority for the SVI interfaces 10 and 20, but the DisSwitch-II sets the default priority (100) for these interfaces.

MD5 authentication is used to avoid rogue VRRP server to become a Master, and thereby, to prevent the Man-in-the-Middle attack.

The following commands execute such configuration. That configuration snippet must be repeated for each VLAN changing the “x” for the corresponding VLAN identifier (10, 20, 30, or 40) and the “P” by the VRRP priority.

```
DisSwitch-I(config)# interface vlan x
DisSwitch-I(config-if-V110)# vrrp x priority P
DisSwitch-I(config-if-V110)# vrrp x ip 192.168.x.254
DisSwitch-I(config-if-V110)# vrrp x authentication ietf-md5 key-string MiKei10!
```

g) NTP Configuration

The NTP server allows time synchronization on all network devices (Server A with IP 172.16.50.1).

```
DisSwitch-I(config)# ntp server 172.16.50.1
DisSwitch-I(config)# clock timezone America/Bogota
DisSwitch-I(config)# ntp source loopback 0
```

h) IP Helper Address Configuration

Server A (172.16.50.1) provides a DHCP server to clients in VLAN 10, 20, and 30. So, the DHCP relay agent has to be enabled on the SVI interfaces (command ip helper-address) for forwarding the DHCP broadcast to the unicast Server A.

```
DisSwitch-I(config)# interface vlan 10
DisSwitch-I(config-if-V110)# ip helper-address 172.16.50.1
DisSwitch-I(config-if-V110)# exit
DisSwitch-I(config)# interface vlan 20
DisSwitch-I(config-if-V120)# ip helper-address 172.16.50.1
DisSwitch-I(config-if-V120)# exit
DisSwitch-I(config)# interface vlan 30
DisSwitch-I(config-if-V130)# ip helper-address 172.16.50.1
DisSwitch-I(config-if-V120)# exit
```

i) DNS Server Configurations

```
DisSwitch-I(config)# ip name-server 172.16.50.1
```

j) Remote Authentication Dial-In User Service (RADIUS) Client Configuration

The RADIUS running on Server A (172.16.50.1) is used to provide centralized authentication to the network devices. The full local user access is necessary if the RADIUS server is not reachable.

Enable local user access.

```
DisSwitch-I(config)# username admin privilege 15 secret cisco
DisSwitch-I(config)# enable secret cisco
```

Configure RADIUS with a secret key. The key will also be configured on the server.

```
DisSwitch-I(config)# radius-server host 172.16.50.1 auth-port 1812 acct-port
DisSwitch-I(config)# radius-server key test123
```

```
DisSwitch-I(config)# ip radius source-interface loopback 0
```

The following login method establishes that RADIUS authentication will be used first, and local user authentication is used instead if Server A is unreachable.

```
DisSwitch-I(config)# aaa authentication login default group radius local  
DisSwitch-I(config)# aaa authentication enable default group radius local
```

Enable authorization for console and exec terminal sessions.

```
DisSwitch-I(config)# aaa authorization console  
DisSwitch-I(config)# aaa authorization exec default group radius local
```

k) Logging Configuration

The level of logging corresponds to notifications (it is equivalent to level 5 and lower).

```
DisSwitch-I(config)# logging trap notifications
```

Configure the Syslog server address where the logs are sent (Server B with IP address 172.16.50.3). The server uses the path /var/log/syslog-ng/10.1.1.6/ to store the log messages.

```
DisSwitch-I(config)# logging host 172.16.50.3
```

The logging source interface is the management interface or loopback.

```
DisSwitch-I(config)# logging source-interface Loopback0
```

l) SNMP Configuration

Enable the Read-only (RO) community string as "emulation" in this particular case.

```
DisSwitch-I(config)# snmp-server community emulation ro
```

Enable the router to send all traps to the Zabbix Server (Server B with IP address 172.16.50.3) with the community string "emulation" and SNMP version 2.

```
DisSwitch-I(config)# snmp-server host 172.16.50.3 version 2c emulation  
DisSwitch-I(config)# snmp-server enable traps
```

m) sFlow Configuration

Send flows to the nProbe server (Server B with IP address 172.16.50.3).

```
DisSwitch-I(config)# sflow destination 172.16.50.3 2055
```

The source interface is the management interface or loopback.

```
DisSwitch-I(config)# sflow source Loopback0
```


Set the sampling properties according to network conditions.

```
DisSwitch-I(config)#sflow sample dangerous 256  
DisSwitch-I(config)#sflow polling-interval 60
```

Enables the flows sending from all active interfaces.

```
DisSwitch-I(config)#sflow run
```

Verify configuration with commands showed in Figure A.3 and the Figure A.4.

```
vEOS-Dis-I#show sflow interfaces  
sFlow Interface (s):  
-----  
Ethernet1 - running  
Ethernet2 - running  
Ethernet3 - running  
Ethernet4 - running  
Ethernet5 - running  
Ethernet6 - running  
vEOS-Dis-I#
```

Figure A.3 Distribution switch command output: show sflow interfaces.

```
vEOS-Dis-I#show sflow  
sFlow Configuration  
-----  
Destination(s):  
  172.16.50.2:2055 ( VRF: default )  
Source(s):  
  10.1.1.6 ( VRF: default )  
  :: ( default ) ( VRF: default )  
Sample Rate: 1048576 ( default )  
Polling Interval (sec): 2.0 ( default )  
Rewrite DSCP value: No  
  
Status  
-----  
Running: Yes  
Polling On: Yes ( default )  
Sampling On: Yes ( default )  
Send Datagrams:  
  Yes ( VRF: default )  
BGP Export:  
  No ( VRF: default )  
Hardware Sample Rate: 1048576 ( default )  
  
Statistics  
-----  
Total Packets: 79  
Number of Samples: 0  
Sample Pool: 0  
Hardware Trigger: 0  
Number of Datagrams: 0  
vEOS-Dis-I#
```

Figure A.4 Distribution switch command output: show sflow.

n) Configuration Files

The `/enterprise_campus/distribution/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration files for the DisSwitch-I and DisSwitch-II devices for consultation purposes.

A.3 Core Layer Configuration

A.3.1 Core Switches Configuration

The two core switches are similarly configured, and the configuration procedure to be followed is as defined for access switches. Therefore, a thorough explanation is not needed, and only the NetFlow configuration is detailed. The configuration files are attached. Both switches are emulated by installed qcow2 images of Cisco vIOS-I2 Qemu appliance (version 15.2).

a) NetFlow Configuration

Send flows to nProbe server (Server B with IP address 172.16.50.3).

```
CoreSwitch-I> enable
CoreSwitch-I# configure terminal
CoreSwitch-I(config)# ip flow-export destination 172.16.50.3 2055
CoreSwitch-I(config)# ip flow-export version 9
```

Enables the flows sending from all active interfaces.

```
CoreSwitch-I(config)# interface GigabitEthernet0/0
CoreSwitch-I(config-if)# ip flow ingress
CoreSwitch-I(config-if)# ip flow egress
CoreSwitch-I(config-if)# exit
```

```
CoreSwitch-I(config)# interface GigabitEthernet0/1
CoreSwitch-I(config-if)# ip flow ingress
CoreSwitch-I(config-if)# ip flow egress
CoreSwitch-I(config-if)# exit
```

```
CoreSwitch-I(config)# interface GigabitEthernet0/2
CoreSwitch-I(config-if)# ip flow ingress
CoreSwitch-I(config-if)# ip flow egress
CoreSwitch-I(config-if)# exit
```

```
CoreSwitch-I(config)# interface GigabitEthernet0/3
CoreSwitch-I(config-if)# ip flow ingress
CoreSwitch-I(config-if)# ip flow egress
CoreSwitch-I(config-if)# exit
```

Verify configuration with following commands.

```
CoreSwitch-I# show ip flow interface
CoreSwitch-I# show ip cache flow
```

b) Configuration Files

The `/enterprise_campus/core/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration files for the CoreSwitch-I and CoreSwitch-II devices for consultation purposes.

A.3.2 Firewall Configuration

The firewall is a Juniper vSRX Qemu appliance on qcow2 disk (version 17.3R1). Next, its configuration is detailed. The configuration files are attached at the end of the explanation.

a) Basic Configuration

Log in as the root user (no password is set).

```
login: root
```

Start the CLI and enter to configuration mode.

```
root#cli
root>configure
root#
```

Configure the root authentication password using a cleartext password (also, it is possible to set an encrypted password or an SSH public key string).

```
[edit]
root# set system root-authentication plain-text-password
New password: Juniper
Retype new password: Juniper
```

Configure the hostname.

```
[edit]
root# set system host-name vSRX-I
root# commit
root@vSRX-I#
```

b) Interfaces Configuration

The interfaces connecting vSRX-I to the Core network elements are named inside0 and inside1. Those relating vSRX-I to the Server Farm are denominated as server0 and server1. The interface connecting vSRX-I to the RouterEDGE router is set with the name outside.

```
[edit]
root@vSRX-I# set interfaces ge-0/0/0 unit 0 family inet address 172.16.0.13/30
root@vSRX-I# set interfaces ge-0/0/0 unit 0 description "Link to CoreSwitch-II"
root@vSRX-I# set interfaces ge-0/0/0 unit 0 alias inside0

root@vSRX-I# set interfaces ge-0/0/1 unit 0 family inet address 172.16.0.9/30
root@vSRX-I# set interfaces ge-0/0/1 unit 0 description "Link to CoreSwitch-I"
root@vSRX-I# set interfaces ge-0/0/1 unit 0 alias inside1

root@vSRX-I# set interfaces ge-0/0/2 unit 0 family inet address 172.16.0.1/30
root@vSRX-I# set interfaces ge-0/0/2 unit 0 description "Link to RouterEDGE"
root@vSRX-I# set interfaces ge-0/0/2 unit 0 alias outside

root@vSRX-I# set interfaces ge-0/0/3 unit 0 family inet address 172.16.0.5/30
root@vSRX-I# set interfaces ge-0/0/3 unit 0 description "Link to ServerSwitch"
root@vSRX-I# set interfaces ge-0/0/3 unit 0 alias server0
```

```
root@vSRX-I# set interfaces ge-0/0/4 unit 0 family inet address 172.16.0.17/30
root@vSRX-I# set interfaces ge-0/0/4 unit 0 description "Link to ServerSwitch"
root@vSRX-I# set interfaces ge-0/0/4 unit 0 alias server1
```

c) OSPF Configuration

Enable routing on the firewall.

```
[edit]
root@vSRX-I# set routing-options router-id 10.1.1.3
```

Configure the default static route.

```
[edit]
root@vSRX-I# edit routing-options
[edit routing-options]
root@vSRX-I# set static route 0.0.0.0/0 next-hop 172.16.0.2
```

Configure the single-area OSPF network.

```
[edit]
root@vSRX-I# set protocols ospf area 0.0.0.0 interface ge-0/0/0
[edit]
root@vSRX-I# set protocols ospf area 0.0.0.0 interface ge-0/0/1
[edit]
root@vSRX-I# set protocols ospf area 0.0.0.0 interface ge-0/0/3
[edit]
root@vSRX-I# set protocols ospf area 0.0.0.0 interface ge-0/0/4
```

Configure the authentication with the MD5 algorithm.

```
[edit]
root@vSRX-I# edit protocols ospf area 0 interface ge-0/0/0.0
[edit protocols ospf area 0.0.0.0 interface ge-0/0/0.0]
root@vSRX-I# set authentication md5 1 key #MyPass!034
[edit protocols ospf area 0.0.0.0 interface ge-0/0/0.0]
root@vSRX-I# top
```

```
[edit]
root@vSRX-I# edit protocols ospf area 0 interface ge-0/0/1.0
[edit protocols ospf area 0.0.0.0 interface ge-0/0/1.0]
root@vSRX-I# set authentication md5 1 key #MyPass!034
[edit protocols ospf area 0.0.0.0 interface ge-0/0/1.0]
root@vSRX-I# top
```

```
[edit]
root@vSRX-I# edit protocols ospf area 0 interface ge-0/0/3.0
[edit protocols ospf area 0.0.0.0 interface ge-0/0/3.0]
root@vSRX-I# set authentication md5 1 key #MyPass!034
[edit protocols ospf area 0.0.0.0 interface ge-0/0/3.0]
root@vSRX-I# top
```

```
[edit]
root@vSRX-I# edit protocols ospf area 0 interface ge-0/0/4.0
[edit protocols ospf area 0.0.0.0 interface ge-0/0/4.0]
root@vSRX-I# set authentication md5 1 key #MyPass!034
[edit protocols ospf area 0.0.0.0 interface ge-0/0/4.0]
root@vSRX-I# top
```

Redistribute static routes with OSPF.

```
[edit]
root@vSRX-I# set policy-options policy-statement exportstatic1 term exportstatic1 from protocol
static
root@vSRX-I# set policy-options policy-statement exportstatic1 term exportstatic1 then accept
root@vSRX-I# set protocols ospf export exportstatic1
```

d) NTP Configuration

The NTP server allows time synchronization on all network devices (Server A with IP 172.16.50.1).

```
[edit]
root@vSRX-I# set system ntp server 172.16.50.1 version 4
root@vSRX-I# set system time-zone America/Bogota
```

e) Logging

Enable logging messages to console and RAM (buffer).

```
[edit]
root@vSRX-I# set system syslog console any info
root@vSRX-I# set system syslog file messages any info
```

Set the Syslog server address where the logs are sent (Server B with IP address 172.16.50.3).

```
[edit]
root@vSRX-I# set system syslog host 172.16.50.3 any any
```

f) Security Zones Configuration

Configure the security zones and bind them to traffic interfaces. Interfaces inside0 and inside1 are binding to the inside_zone zone. Interfaces server0, and server1 are binding to the server_zone zone. Interface outside is binding to outside_zone.

```
[edit]
root@vSRX-I# set security zones security-zone inside_zone interfaces ge-0/0/0
root@vSRX-I# set security zones security-zone inside_zone interfaces ge-0/0/1
root@vSRX-I# set security zones security-zone server_zone interfaces ge-0/0/3
root@vSRX-I# set security zones security-zone server_zone interfaces ge-0/0/4
root@vSRX-I# set security zones security-zone outside_zone interface ge-0/0/2
```

Enable all expected host-inbound traffic from the inside_zone zone (ping, ssh, telnet, ospf).

```
root@vSRX-I# set security zones security-zone inside_zone host-inbound-traffic system-services ping
```

```

root@vSRX-I# set security zones security-zone inside_zone host-inbound-traffic system-services ssh
root@vSRX-I# set security zones security-zone inside_zone host-inbound-traffic system-services
telnet
root@vSRX-I# set security zones security-zone inside_zone host-inbound-traffic protocols ospf
root@vSRX-I# set security zones security-zone inside_zone host-inbound-traffic protocols ospf3

```

Enable all expected host-inbound traffic from the server_zone zone (ping, ssh, telnet, ntp, ospf).

```

root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic system-services ping
root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic system-services ssh
root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic system-services
telnet
root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic system-services ntp
root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic protocols ospf
root@vSRX-I# set security zones security-zone server_zone host-inbound-traffic protocols ospf3

```

Enable all expected host-inbound traffic from the outside_zone zone (ospf).

```

root@vSRX-I# set security zones security-zone outside_zone host-inbound-traffic protocols ospf
root@vSRX-I# set security zones security-zone outside_zone host-inbound-traffic protocols ospf3

```

g) Address Book Configuration

The address book inside_addresses contains the end VLANs network addresses (which conforms an address set) and the loopbacks addresses of the access, distribution, and core switches.

```

root@vSRX-I# set security address-book inside_addresses address loopbacks_inside 10.1.1.0/24
root@vSRX-I# set security address-book inside_addresses address vlan10_192.168.10
192.168.10.0/24
root@vSRX-I# set security address-book inside_addresses address vlan20_192.168.20
192.168.20.0/24
root@vSRX-I# set security address-book inside_addresses address vlan30_192.168.30
192.168.30.0/24
root@vSRX-I# set security address-book inside_addresses address vlan40_192.168.40_mgmt
192.168.40.0/24

root@vSRX-I# set security address-book inside_addresses address-set end_vlans address
vlan10_192.168.10
root@vSRX-I# set security address-book inside_addresses address-set end_vlans address
vlan20_192.168.20
root@vSRX-I# set security address-book inside_addresses address-set end_vlans address
vlan30_192.168.30
root@vSRX-I# set security address-book inside_addresses address-set end_vlans address
vlan40_192.168.40_mgmt

root@vSRX-I# set security address-book inside_addresses attach zone inside_zone

```

The address book outside_addresses contains the DNSs addresses (which conforms an address set) and the loopback address of the RouterEDGE router.

```

root@vSRX-I# set security address-book outside_addresses address google_dns1 8.8.8.8
root@vSRX-I# set security address-book outside_addresses address google_dns2 8.8.4.4
root@vSRX-I# set security address-book outside_addresses address-set google_dns address
google_dns1

```

```

root@vSRX-I# set security address-book outside_addresses address-set google_dns address
google_dns2
root@vSRX-I# set security address-book outside_addresses address loopback_edge 10.1.1.5
root@vSRX-I# set security address-book outside_addresses attach zone outside_zone

```

The address book datacenter_addresses contains the servers addresses and the VLAN5 network address.

```

root@vSRX-I# set security address-book datacenter_addresses address vlan50_172.16.50
172.16.50.0/24
root@vSRX-I# set security address-book datacenter_addresses address server_a 172.16.50.1
root@vSRX-I# set security address-book datacenter_addresses address server_b 172.16.50.3
root@vSRX-I# set security address-book datacenter_addresses attach zone server_zone

```

h) Application Sets Configuration

```

root@vSRX-I# set applications application-set server_tcp_out application junos-http
root@vSRX-I# set applications application-set server_tcp_out application junos-https
root@vSRX-I# set applications application-set server_tcp_out application junos-dns-tcp

root@vSRX-I# set applications application-set server_udp_out application junos-dns-udp

root@vSRX-I# set applications application snmp protocol udp
root@vSRX-I# set applications application snmp destination-port 161

root@vSRX-I# set applications application snmptrap protocol udp
root@vSRX-I# set applications application snmptrap destination-port 162

root@vSRX-I# set applications application netflow protocol udp
root@vSRX-I# set applications application netflow destination-port 2055

root@vSRX-I# set applications application dhcp protocol udp
root@vSRX-I# set applications application dhcp source-port 67 destination-port 67

root@vSRX-I# set applications application mysql protocol tcp
root@vSRX-I# set applications application mysql destination-port 3306

```

i) Traffic Rules Configuration

Permit ping between network elements.

```

[edit]
root@vSRX-I# set security policies global policy policy_ping match application junos-ping source-
address any destination-address any
root@vSRX-I# set security policies global policy policy_ping then permit

```

Permit HTTP, HTTPS, and DNS traffic.

```

root@vSRX-I# set security policies global policy policy_tcp_udp match application server_tcp_out
application server_udp_out from-zone any to-zone any destination-address any source-address any
root@vSRX-I# set security policies global policy policy_tcp_udp then permit

```

Permit SSH access from management VLAN to servers.

```
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_ssh
match application junos-ssh source-address vlan40_192.168.40_mgmt destination-address
vlan50_172.16.50
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_ssh
then permit
```

Permit MySQL traffic between servers and the management VLAN.

```
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy
policy_databases match application mysql source-address vlan40_192.168.40_mgmt destination-
address vlan50_172.16.50
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy
policy_databases then permit
```

Permit Syslog traps to ServerB from inside_zone and outside_zone to server_zone.

```
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_syslog
match application junos-syslog source-address any destination-address server_b
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_syslog
then permit
root@vSRX-I# set security policies from-zone outside_zone to-zone server_zone policy
policy_syslog match application junos-syslog source-address any destination-address server_b
root@vSRX-I# set security policies from-zone outside_zone to-zone server_zone policy
policy_syslog then permit
```

Permit NTP requests.

```
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_ntp
match application junos-ntp source-address loopbacks_inside destination-address server_a
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_ntp
then permit

root@vSRX-I# set security policies from-zone outside_zone to-zone server_zone policy policy_ntp
match application junos-ntp source-address any destination-address any
root@vSRX-I# set security policies from-zone outside_zone to-zone server_zone policy policy_ntp
then permit

root@vSRX-I# set security policies from-zone server_zone to-zone outside_zone policy policy_ntp
match application junos-ntp source-address server_a destination-address any
root@vSRX-I# set security policies from-zone server_zone to-zone outside_zone policy policy_ntp
then permit
```

Permit SNMP probes and traps between inside_zone and server_zone.

```
root@vSRX-I# set security policies global policy policy_snmp match application snmp from-zone
server_zone from-zone inside_zone to-zone server_zone to-zone inside_zone destination-address any
source-address any
root@vSRX-I# set security policies global policy policy_snmp then permit

root@vSRX-I# set security policies global policy policy_snmptrap match application snmptrap from-
zone inside_zone to-zone server_zone destination-address any source-address any
root@vSRX-I# set security policies global policy policy_snmptrap then permit
```


Permit NetFlow/sFlow messages from loopbacks to the servers (VLAN5).

```
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_netflow
match application netflow source-address loopbacks_inside destination-address vlan50_172.16.50
root@vSRX-I# set security policies from-zone inside_zone to-zone server_zone policy policy_netflow
then permit
```

Permit DHCP traffic between inside_zone and server_zone.

```
root@vSRX-I# set security policies global policy policy_dhcp match application dhcp from-zone
inside_zone from-zone server_zone to-zone server_zone to-zone inside_zone destination-address any
source-address any
root@vSRX-I# set security policies global policy policy_dhcp then permit
```

j) Verify the Configuration

```
[edit]
root@# commit check
configuration check succeed
```

k) Commit the Configuration

```
[edit]
root@# commit
commit complete
```

l) Configuration File

The `/enterprise_campus/core/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration file for the vSRX firewall device for consultation purposes.

A.4 Server Farm Configuration

A.4.1 Switch Configuration

The switch is a Cisco vIOS-I2 Qemu appliance on qcow2 disks (version 15.2). The configuration procedure to be followed is as defined for the access and core switches. Therefore, a thorough explanation is not needed, and only the configuration file is attached. The `/enterprise_campus/server_farm/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration file for consultation purposes.

A.4.2 Server A Configuration

A.4.2.1 Main configuration

The ServerA is an Ubuntu 16.04.4 LTS Vmware virtual machine. The first step to configure it is to update the list of available packages and install newer versions of the current packages.

```
$ sudo su
# apt-get update && apt-get upgrade
```

a) Basic Configuration

Change the hostname.

```
# vi /etc/hostname
```

```
ServerA
```

Add the hostname with an IP address 127.0.1.1 into the file `/etc/hosts` for hostname resolution purposes, and reboot to apply changes.

```
# echo "127.0.1.1 ServerA" >> /etc/hosts
```

Configure a banner for SSH connections.

```
# vi /etc/issue.net
```

```
Bienvenido al servidor "ServerA"
Todas las conexiones estan siendo monitorizadas y grabadas.
Finalice su conexión si no es un usuario autorizado.
```

Install the OpenSSH server and edit its configuration file by uncommenting the line starting with `#Banner` as follows.

```
# apt-get install openssh-server
# vi /etc/ssh/sshd_config
```

```
Banner /etc/issue.net
```

Restart the SSH service.

```
# systemctl restart ssh
```

b) VGA Output to Serial Port Redirection

GNS3 supports the console connection via a serial port connection with the appliances. ServerA is a Vmware virtual machine running inside GNS3. So, to ensure the connection through the GNS3 platform, it is necessary to reconfigure Ubuntu for redirecting the VGA output to a serial port.

```
$ sudo su
# vi /lib/systemd/system/ttyS0.service
```

```
[Unit]
Description=Serial Console Service
[Service]
ExecStart=/sbin/getty -L 115200 ttyS0 vt102
Restart=always
[Install]
WantedBy=multi-user.target
```

Enable and start ttyS0 service. Then reboot the ServerA.

```
# systemctl daemon-reload
# systemctl enable ttyS0
# systemctl start ttyS0
```

c) Interface Configuration

Edit the configuration file `/etc/network/interfaces` to assign a static IP address to the network interface.

```
$ sudo vi /etc/network/interfaces
```

```
#The primary network interface
auto ens33
iface ens33 inet static
address 172.16.50.1
gateway 172.16.50.254
netmask 255.255.255.0
broadcast 172.16.50.255
dns_nameservers 172.16.50.1
```

Restart the network interfaces to apply the changes.

```
$ sudo /etc/init.d/networking restart
```

A.4.2.2 DNS server installation and configuration

a) Bind Server Installation

```
$ sudo apt-get update
# sudo apt-get install bind9 bind9-doc
```

b) Bind Server Configuration

Edit the configuration file `/etc/bind/named.conf.options` to define the allowed client list to access the DNS server and the public name servers for resolving domains.

```
# vi /etc/bind/named.conf.options
```

First, add the following access list of clients allowed before the options block.

```
acl ourclients {  
    10.0.0.0/24;  
    10.1.1.0/24;  
    192.168.10.0/24;  
    192.168.20.0/24;  
    192.168.30.0/24;  
    192.168.40.0/24;  
    172.16.0.0/24;  
    172.16.50.0/24;  
    localhost;  
    localnets;  
};
```

Inside the options block, reference to the access list of clients and allow recursion as following.

```
recursion yes;  
allow-query {  
    ourclients;  
};
```

Set Google public DNS servers as forwarders inside options block. The line “forward only” makes the Bind server forward the request to Google DNS servers.

```
forwarders {  
    8.8.8.8;  
    8.8.4.4;  
};  
  
forward only;
```

Change the line dnssec-validation and add the new line dnssec-enable as following.

```
dnssec-enable yes;  
dnssec-validation yes;
```

Save the file and check the configuration with the next command. If there are no errors, the output is blank.

```
# named-checkconf
```

c) Logging Configuration

Create the directory */var/log/named* and assign Bind as its owner and group.

```
# mkdir /var/log/named/  
# chown bind:bind /var/log/named/
```

Edit the configuration file */etc/bind/named.conf.options* adding the following lines.

```
logging {  
  channel query_log {  
    file "/var/log/named/bind.log" versions 3 size 5m;  
    severity info;  
    print-time yes;  
    print-severity yes;  
    print-category yes;  
  };  
  category queries {  
    query_log;  
  };  
};
```

Inside the options block, add the following line.

```
querylog yes;
```

Save the file and check the configuration. Then, restart Bind server and all DNS queries will be stored in the file */var/log/named/bind.log*

```
# named-checkconf  
# systemctl restart bind9
```

d) DNS Configuration File

The `/enterprise_campus/server_farm/ServerA/DNS_server/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration file *named.conf.options* for consultation purposes.

A.4.2.3 NTP server installation and configuration

a) NTP Installation

```
$ sudo apt-get install ntp
```

b) NTP Client Configuration

Edit the configuration file */etc/ntp.conf* to specifying the NTP public servers geographically near and commenting the lines of the preconfigured NTP servers as following.

```
$ sudo su  
# vi /etc/ntp.conf
```

```
#pool 0.ubuntu.pool.ntp.org iburst
#pool 1.ubuntu.pool.ntp.org iburst
#pool 2.ubuntu.pool.ntp.org iburst
#pool 3.ubuntu.pool.ntp.org iburst
#pool ntp.ubuntu.com

# Add the following lines.
server 0.co.pool.ntp.org
server 1.south-america.pool.ntp.org
server 2.pool.ntp.org
```

Add the following lines to allow synchronization with a public NTP server and restrict the access of those servers to the local NTP server.

```
restrict -4 default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
```

Add the following lines to allow queries from the emulated network elements.

```
restrict 10.1.1.0 mask 255.255.255.0 nomodify notrap
restrict 172.16.0.0 mask 255.255.255.0 nomodify notrap
restrict 10.0.0.0 mask 255.0.0.0 nomodify notrap
restrict 192.168.0.0 mask 255.255.0.0 nomodify notrap
```

Restart the NTP service.

```
# systemctl restart ntp
```

c) NTP Configuration File

The `/enterprise_campus/server_farm/ServerA/NTP_server/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration file `ntp.conf` for consultation purposes.

A.4.2.4 DHCP server installation and configuration

a) DHCP Server Installation

```
$ sudo apt-get install isc-dhcp-server
```

b) DHCP Server Configuration

Edit the configuration file `/etc/dhcp/dhcpd.conf` to define the addresses ranges for each subnetwork (VLAN 1, 2, 3, and 5).

```
# vi /etc/dhcp/dhcpd.conf
```

```
option domain-name "mycompany.sk";

default-lease-time 600;
max-lease-time 7200;

subnet 172.16.50.0 netmask 255.255.255.0 {
    range                172.16.50.1 172.16.50.240;
    option routers        172.16.50.254;
    option subnet-mask    255.255.255.0;
    option broadcast-address 172.16.50.255;
    option domain-name-servers 172.16.50.1;
    option ntp-servers    172.16.50.1;
}

subnet 192.168.10.0 netmask 255.255.255.0 {
    range                192.168.10.1 192.168.10.240;
    option routers        192.168.10.254;
    option subnet-mask    255.255.255.0;
    option broadcast-address 192.168.10.255;
    option domain-name-servers 172.16.50.1;
    option ntp-servers    172.16.50.1;
}

subnet 192.168.20.0 netmask 255.255.255.0 {
    range                192.168.20.1 192.168.20.240;
    option routers        192.168.20.254;
    option subnet-mask    255.255.255.0;
    option broadcast-address 192.168.20.255;
    option domain-name-servers 172.16.50.1;
    option ntp-servers    172.16.50.1;
}

subnet 192.168.30.0 netmask 255.255.255.0 {
    range                192.168.30.1 192.168.30.240;
    option routers        192.168.30.254;
    option subnet-mask    255.255.255.0;
    option broadcast-address 192.168.30.255;
    option domain-name-servers 172.16.50.1;
    option ntp-servers    172.16.50.1;
}
```

Configure the listening interface and the configuration file path in the file */etc/default/isc-dhcp-server*.

```
# vi /etc/default/isc-dhcp-server
```

```
DHCPD_CONF=/etc/dhcp/dhcpd.conf
INTERFACES="ens33"
```

Enable and start the DHCP server. The file */var/lib/dhcp/dhcpd.leases* stores the leased addresses.

```
# sudo systemctl enable isc-dhcp-server
# sudo systemctl start isc-dhcp-server
```

A.4.2.5 RADIUS installation and configuration

a) FreeRADIUS Installation

```
$ sudo apt-get install freeradius
```

Edit the configuration file `/etc/freeradius/clients.conf` to define the clients.

```
# vi /etc/freeradius/clients.conf
```

The `/enterprise_campus/server_farm/ServerA/FreeRADIUS_server/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the complete configuration file `clients.conf` for consultation purposes.

Edit the configuration file `/etc/freeradius/users` as following.

```
# vi /etc/freeradius/users

# User privilege level 1
raadmin Cleartext-Password := "racisco"
Service-Type = NAS-Prompt-User,
# User privilege level 15
raadmin15 Cleartext-Password := "racisco15"
Service-Type = NAS-Prompt-User,
cisco-avpair = "shell:priv-lvl=15"
# Enable password
$enab15$ Cleartext-Password := "racisco"
Service-Type = NAS-Prompt-User,
```

The `/enterprise_campus/server_farm/ServerA/FreeRADIUS_server/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the complete configuration file `users` for consultation purposes.

Edit the configuration file `/etc/freeradius/radiusd.conf` to log authentication requests as following. The file `/var/log/freeradius/radius.log` stores the logs.

```
# vi /etc/freeradius/radiusd.conf

auth = yes
auth_badpass = yes
auth_goodpass = yes
```

Restart FreeRADIUS.

```
# systemctl restart freeradius
```


A.4.3 Server B Configuration

The ServerB is the Zabbix Appliance v3.4.0, which is pre-installed and pre-configured Zabbix software over Ubuntu 16.04 virtual machine.

A.4.3.1 SNMP server configuration (Zabbix)

We use the Zabbix Appliance for trouble-free deployment. Zabbix Appliance virtual machine has prepared the Zabbix server with MySQL support, all monitoring capabilities, including the configuration for handle SNMP traps (download URL: https://www.zabbix.com/download_appliance). Nevertheless, it also is possible to install it from the distribution packages or download the container.

a) IP Address Configuration

By default, the appliance uses DHCP to obtain the IP address. This configuration is used in this work, so the ServerA assigns the IP address for 172.16.50.0/24 network (172.16.50.3 has been assigned).

b) Time Zone Configuration

To set the correct time zone, copy the appropriate file from `/usr/share/zoneinfo/Americas/Bogota` to `/etc/localtime`.

```
$ cp /usr/share/zoneinfo/Americas/Bogota /etc/localtime
```

Frontend timezone should also be modified in `/etc/apache2/conf-available/zabbix.conf`.

c) Credentials

Table A.2 Credentials for Zabbix appliance.

	Login/User	Password
Operative System	appliance	zabbix
Database	root	<random password>
	zabbix	<random password>
Frontend	Admin	zabbix

As shown in Table A.2, database passwords are randomly generated and stored in the file `/root/.my.cnf`. The following files have to be edited to change the database user password if necessary.

```
/etc/zabbix/zabbix_server.conf  
/etc/zabbix/web/zabbix.conf.php
```

To access the Zabbix frontend, open a browser with the URL <http://172.16.50.3/zabbix> from a PC of VLAN4.

A.4.3.2 Syslog server installation and configuration (Rsyslog)

There are different Syslog server implementations; in this case, two were used. First, **syslog-ng** to save the logs in files. Second, **Rsyslog** to store logs in a database. The LogAnalyzer tool, configured for Rsyslog, is used to monitor the stored logs via Web.

a) Syslog-ng Installation and Configuration

Install the package syslog-ng.

```
$ sudo apt-get install syslog-ng
$ sudo su
```

Edit the configuration file `/etc/syslog-ng/conf.d/firewalls.conf` as following.

```
# cd /etc/syslog-ng/conf.d
# vi firewalls.conf

options {
    create_dirs(yes);
    owner(ubuntu);
    group(ubuntu);
    perm(0640);
    dir_owner(ubuntu);
    dir_group(ubuntu);
    dir_perm(0750);
};

source s_net {
    tcp(ip(0.0.0.0) port(514));
    udp(ip(0.0.0.0) port(514));
};

destination d_host-specific {
    file("/var/log/syslog-ng/$HOST/$YEAR/$MONTH/$HOST-$YEAR-$MONTH-$DAY.log");
};

log {
    source(s_net);
    destination(d_host-specific);
};
```

Restart the Syslog server. This server will store the logs in the directory `/var/log/syslog-ng/[source device IP]/[year]/[month]/[day]`

```
# service syslog-ng restart
```

b) Rsyslog Installation and Configuration

The first step is to install the MySQL server on Ubuntu and, once installed, follow the next instructions. In this case, the Zabbix appliance guarantees to have MySQL installed.

Install the following key.

```
$ sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com AEF0CF8E
```

Edit the configuration file /etc/apt/sources.list as following.

```
$ sudo vi /etc/apt/sources.list
```

```
deb http://ubuntu.adiscon.com/v7-devel precise/  
deb-src http://ubuntu.adiscon.com/v7-devel precise/
```

Save the file and update the system.

```
$ sudo apt-get update && sudo apt-get upgrade
```

Install Rsyslog with the following command.

```
$ sudo apt-get install rsyslog
```

Install the Rsyslog support for MySQL databases.

```
# install rsyslog-mysql
```

The installation wizard will be open and will ask if you want to configure the database for rsyslog-mysql with dbconfig-common. Select "Yes" and then press ENTER.

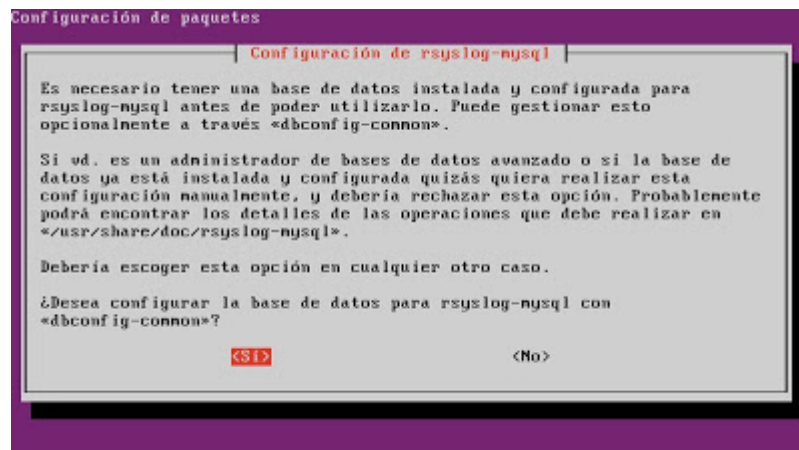


Figure A.5 rsyslog-mysql configuration (step 1).

Enter the password of MySQL Server root superuser (see credentials of A.4.3.1 section). The rsyslog-mysql package must access MySQL and create a user, a catalog, and the corresponding tables.

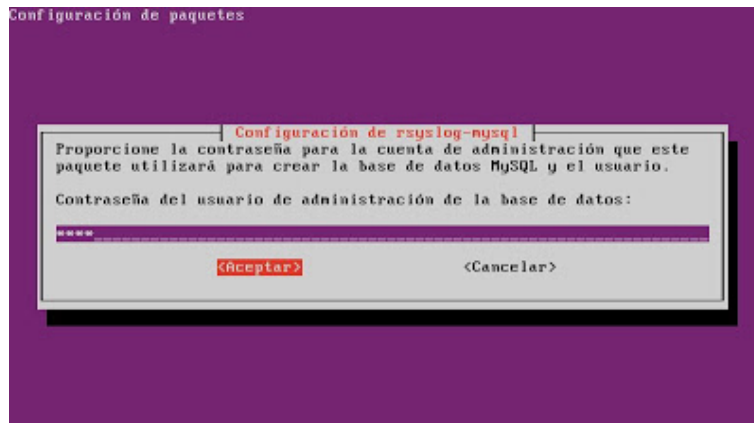


Figure A.6 rsyslog-mysql configuration (step 2).

Enter the password for rsyslog-mysql to access the MySQL server.

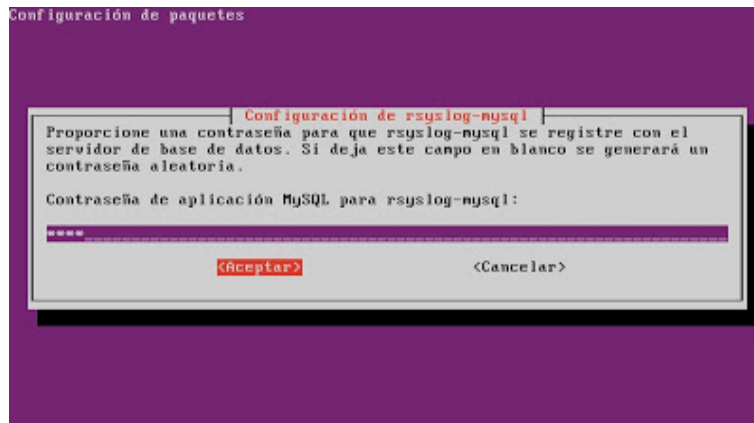


Figure A.7 rsyslog-mysql configuration (step 3).

As a result, the necessary configuration files, the user, catalog, and tables will be created in the MySQL database. MySQL Workbench now will show the "Syslog" scheme created by rsyslog-mysql, and the tables "SystemEvents" and "SystemEventsProperties".

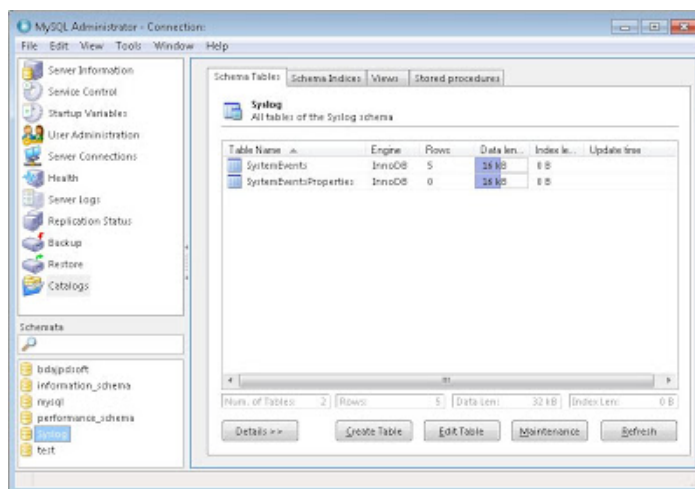


Figure A.8 Syslog scheme in MySQL workbench.

MySQL Workbench also will show the "rsyslog" user.

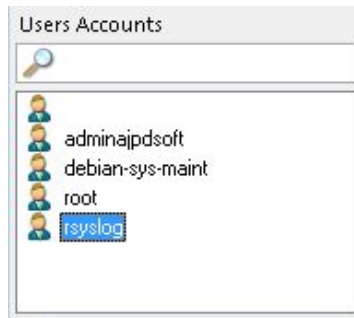


Figure A.9 rsyslog user in MySQL workbench.

Once the module for MySQL is installed, edit the file `/etc/rsyslog.conf` to enable RSyslog listening on ports 514/UDP and 514/TCP. Search the next lines and uncomment them as following.

```
# vi /etc/rsyslog.conf
```

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

c) LogAnalyzer Installation and Configuration

The first step is to install PHP5 y Apache2 on Ubuntu. Any Apache2 installation guide found on the web is useful. To install PHP5, see the following steps.

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:ondrej/php
$ sudo apt-get update
$ sudo apt-get install php5.6
```

Add MySQL support to PHP.

```
$ sudo apt-get install php5.6-mysql
```

Once installed PHP, create a folder for the temporary download of the compressed LogAnalyzer installation package.

```
$ sudo mkdir /tmp/loganalyzer
```

Access the folder created above, and download the LogAnalyzer installation package (the version used is loganalyzer-4.1.7).

```
$ cd /tmp/loganalyzer
$ wget http://download.adiscon.com/loganalyzer/loganalyzer-4.1.7.tar.gz
```

Unzip the package and access the created folder.

```
$ tar -xvf loganalyzer-4.1.7.tar.gz
$ cd /tmp/loganalyzer/loganalyzer-4.1.7
```

Create the loganalyzer subfolder inside /var/www/html/ and copy the contents of the "src" folder within it. Repeat the process for the "contrib" folder.

```
$ sudo mkdir /var/www/html/loganalyzer
$ sudo cp -R /tmp/loganalyzer/loganalyzer-4.1.7/src/* /var/www/html/loganalyzer
$ sudo cp -R /tmp/loganalyzer/loganalyzer-4.1.7/contrib/* /var/www/html/loganalyzer
```

Set write permissions for configure.sh and secure.sh files.

```
$ cd /var/www/html/loganalyzer
$ sudo chmod +x configure.sh secure.sh
```

Start LogAnalyzer via web from a PC of VLAN4 by opening a browser with the URL <http://172.16.50.3/loganalyzer>.

LogAnalyzer will start. At the first start, it will give an error message because it detects that it has not been configured. Click on "here" to install Adiscon LogAnalyzer.



Figure A.10 LogAnalyzer installation (error at the first start).

The wizard will start. Press "Next" in the first step.

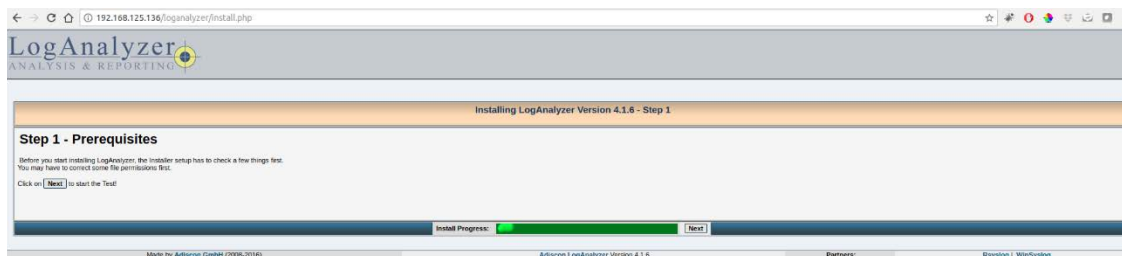


Figure A.11 LogAnalyzer installation (step 1).

In the second step, the LogAnalyzer installation wizard will check if config.php file is writable. Press "Next".

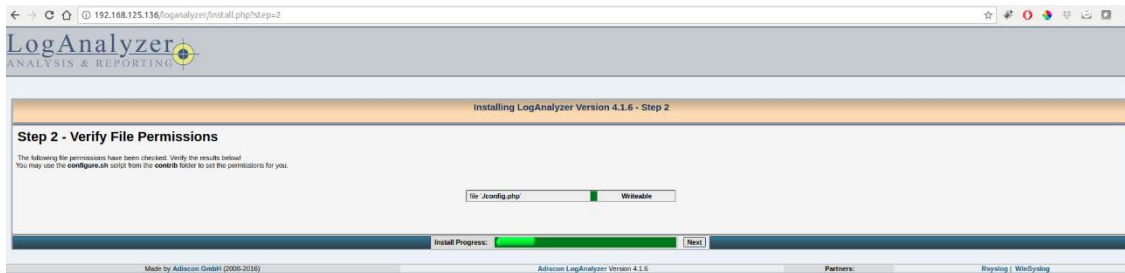


Figure A.12 LogAnalyzer installation (step 2).

In the third step, enter the configuration data as following.

- Enable User database:** Yes
- Database Host:** localhost
- Database Port:** 3306
- Database Name:** loganalyzer (MySQL catalog name to be created for LogAnalyzer).
- Table prefix:** logcon_ (it is the prefix added to the names of the tables).
- Database User:** root
- Database Password:** (password for the root user. See credentials of A.4.3.1 section).
- Require user to be logged in:** Yes
- Authentication method:** Internal authentication.

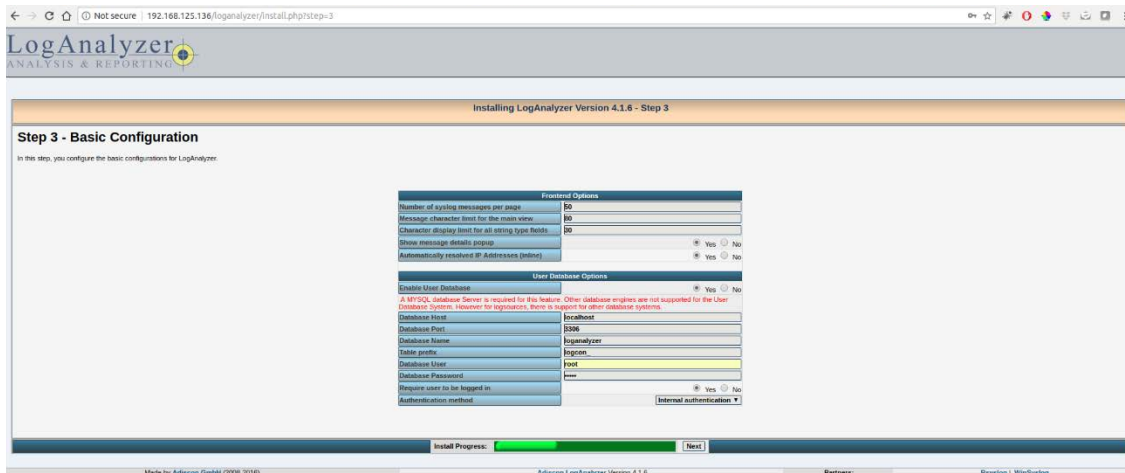


Figure A.13 LogAnalyzer installation (step 3).

In the fourth and fifth step, press "Next".



Figure A.14 LogAnalyzer installation (step 4).

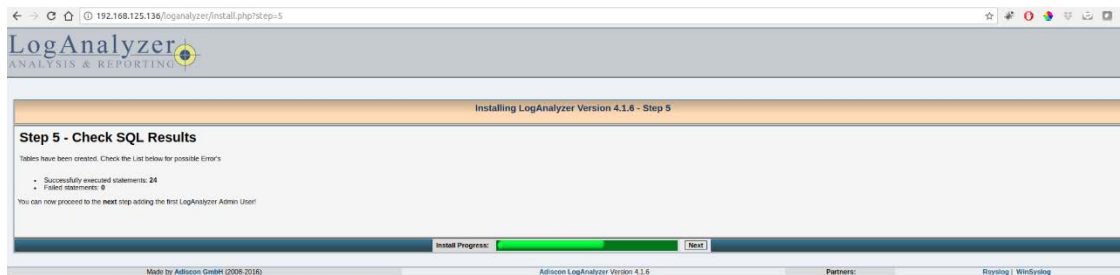


Figure A.15 LogAnalyzer installation (step 5).

In the sixth step, enter the user data used to access LogAnalyzer.

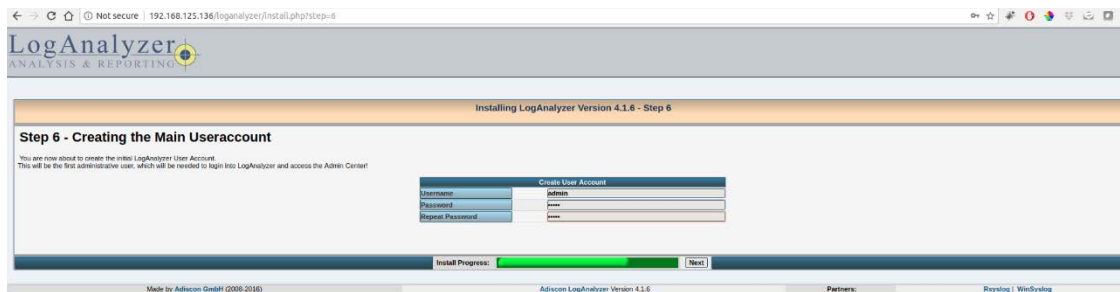


Figure A.16 LogAnalyzer installation (step 6).

In the seventh step, enter the Rsyslog data as a data source.

- Name of the Source:** Syslog emulated network
- Source Type:** MYSQL Native
- Select View:** Syslog Fields
- Table type:** MonitorWare
- Database Host:** localhost
- Database Name:** Syslog
- Database Tablename:** SystemEvents
- Database User:** rsyslog (or root)
- Database Password:** (password for database user).
- Enable Row Counting:** No (because there will be hundreds of thousands of events. Option "Yes" is not advisable because it can slow down the access).

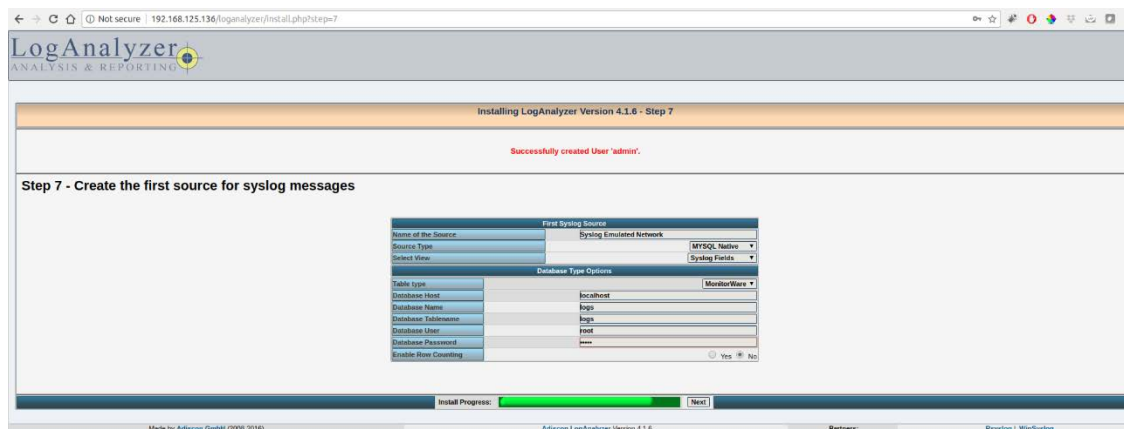


Figure A.17 LogAnalyzer installation (step 7).

Now it is possible to view the logs of the network elements with the LogAnalyzer URL used previously.

A.4.3.3 NetFlow collector installation and configuration (nProbe)

nProbe is selected as Netflow/sflow server collector. To install nProbe, download the stable build packages as follows.

```
$ wget http://apt-stable.ntop.org/16.04/all/apt-ntop-stable.deb
$ dpkg -i apt-ntop-stable.deb
```

Install nProbe with the following commands.

```
$ sudo apt-get clean all
$ sudo apt-get update
$ sudo apt-get install pfring nprobe ntopng ntopng-data n2disk cento nbox
```

To collect incoming flows, use the following command.

```
$ nprobe --zmq "tcp://*:5556" -i none -n none --collector-port 2055 --dump-path [flow_storage_path]
--timestamp-format 2 -T "[flow_parameters]"
```

Where `flow_storage_path` is the directory path that will store the flows, and `flow_parameters` is the flow descriptors list to be saved. `flow_parameters` should be listed as follows: "%param1 %param2 ... \$paramN". The flows will be stored inside the directory tree `[flow_storage_path]/[year]/[month]/[day]/[hour]/[minute].flow`

A.5 Enterprise Edge and Service Provider Configuration

A.5.1 Router EDGE Configuration

The RouterEDGE is a Cisco IOSv Qemu appliance on qcow2 disk (version 15.6(2)T). Next, its configuration is detailed. The configuration file is attached at the end of the explanation.

a) Basic Configuration

Change the hostname of the router and create a local user with a password.

```
Router> en
Router# conf t
Router(config)# hostname RouterEDGE
RouterEDGE(config)# username admin secret cisco
RouterEDGE(config)# enable secret cisco
```

b) IP Addresses Configuration

```
RouterEDGE(config)# interface GigabitEthernet 0/2
RouterEDGE(config-if)# description Link to vSRX-I
RouterEDGE(config-if)# ip address 172.16.0.2 255.255.255.252
RouterEDGE(config-if)# no shutdown
RouterEDGE(config-if)# exit

RouterEDGE(config)# interface GigabitEthernet 0/1
RouterEDGE(config-if)# description Link to ISP1
RouterEDGE(config-if)# ip address 198.10.10.2 255.255.255.252
RouterEDGE(config-if)# no shutdown
RouterEDGE(config-if)# exit

RouterEDGE(config)# interface GigabitEthernet 0/3
RouterEDGE(config-if)# description Link to ISP2
RouterEDGE(config-if)# ip address 197.10.10.2 255.255.255.252
RouterEDGE(config-if)# no shutdown
RouterEDGE(config-if)# exit

RouterEDGE(config)# interface loopback 0
RouterEDGE(config-if)# description Management
RouterEDGE(config-if)# ip address 10.1.1.5 255.255.255.255
RouterEDGE(config-if)# no shutdown
RouterEDGE(config-if)# exit
```

c) NAT Configuration

In the standard access-list 1 define the subnetworks that can be translated.

```
RouterEDGE(config)# access-list 1 permit 192.168.10.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 192.168.20.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 192.168.30.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 192.168.40.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 10.0.0.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 10.1.1.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 172.16.0.0 0.0.0.255
RouterEDGE(config)# access-list 1 permit 172.16.50.0 0.0.0.255
RouterEDGE(config)# access-list 1 deny any
```

We are assuming that the prefix 195.1.1.0/24 is assigned to the enterprise network. The address pool 195.1.1.128/25 is used to translate addresses from campus network and server farm to public addresses.

```
RouterEDGE(config)# ip nat pool 1 195.1.1.1 195.1.1.127 netmask 255.255.255.128
```

Configure NAT translation by relating the inside networks with the corresponding pool.

```
RouterEDGE(config)# ip nat inside source list 1 pool 1 overload
```

Define inside and outside interfaces.

```
RouterEDGE(config)# interface GigabitEthernet 0/1
RouterEDGE(config-if)# ip nat outside
```

```
RouterEDGE(config)# interface GigabitEthernet 0/2
RouterEDGE(config-if)# ip nat inside
```

```
RouterEDGE(config)# interface gigabitEthernet 0/2
RouterEDGE(config-if)# ip nat outside
```

d) Static Routes Configuration

```
RouterEDGE(config)# ip route 172.16.0.0 255.255.0.0 172.16.0.1
RouterEDGE(config)# ip route 192.168.0.0 255.255.192.0 172.16.0.1
RouterEDGE(config)# ip route 10.0.0.0 255.0.0.0 172.16.0.1
```

e) BGP Configuration

Configure a static route to 195.1.1.0/24, pointing to a null interface.

```
RouterEDGE(config)# ip route 195.1.1.0 255.255.255.0 null0
```

Configure the autonomous system number for RouterEDGE, assuming that 64500 is the number assigned to the enterprise and 64501 to the ISPs. Also, configure authentication between peers.

```
RouterEDGE(config)# router bgp 64500
RouterEDGE(config-router)# neighbor 198.10.10.1 remote-as 64501
RouterEDGE(config-router)# neighbor 198.10.10.1 password isp1md5pass
RouterEDGE(config-router)# neighbor 197.10.10.1 remote-as 64502
RouterEDGE(config-router)# neighbor 197.10.10.1 password isp2md5pass
RouterEDGE(config-router)# network 195.1.1.0 mask 255.255.255.0
RouterEDGE(config-router)# neighbor isp-group peer-group
RouterEDGE(config-router)# neighbor isp-group ttl-security hops 1
RouterEDGE(config-router)# neighbor isp-group filter-list 10 out
RouterEDGE(config-router)# neighbor 198.10.10.1 peer-group isp-group
RouterEDGE(config-router)# neighbor 198.10.10.1 route-map setlocalin in
RouterEDGE(config-router)# neighbor 197.10.10.1 peer-group isp-group
RouterEDGE(config)# ip as-path access-list 10 permit ^$
```

The preferred way to connect the Internet is ISP1, so configure a local preference 150 for prefixes from ISP1, while maintaining the default local preference (100) for prefixes from ISP2.

```
RouterEDGE(config)# route-map setlocalin permit 10
RouterEDGE(config-route-map)# set local-preference 150
```

f) NTP Configuration

```
RouterEDGE(config)# ntp server 172.16.50.1
RouterEDGE(config)# clock timezone mytime -5 0
RouterEDGE(config)# ntp source loopback 0
```

g) Logging Configuration

```
RouterEDGE(config)# logging trap notifications
RouterEDGE(config)# logging host 172.16.50.3
RouterEDGE(config)# logging source-interface loopback 0
```

h) DNS Configuration

```
RouterEDGE(config)# ip name-server 8.8.8.8 8.8.4.4
RouterEDGE(config)# ip domain-lookup
```

i) SSH Access and VTY Access-list Configuration

```
RouterEDGE(config)# ip domain name companyXYZ.sk
RouterEDGE(config)# ip ssh version 2
RouterEDGE(config)# crypto key generate rsa modulus 4096
RouterEDGE(config)# line vty 0 924
RouterEDGE(config-line)# login local
RouterEDGE(config-line)# transport input ssh
```

Create an access-list that permits login to VTY from the VLAN4 only.

```
RouterEDGE(config)# ip access-list standard ssh-access
RouterEDGE(config-std-nacl)# permit 192.168.40.0 0.0.0.255
RouterEDGE(config-std-nacl)# deny any
RouterEDGE(config)# line vty 0 924
RouterEDGE(config-line)# access-class ssh-access in
```

j) Configuration file

The /edge/ directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration file for the RouterEDGE device for consultation purposes.

A.5.2 ISP Configuration

The two internet service providers are similarly configured. Therefore, the configuration for one service provider (ISP1) is indicated, and both configuration files are attached to perform the same procedure with the remaining ISP. Both routers are Cisco 7200 routers (IOS version 15.2(4)M7).

a) IP Address Configuration

ISP1 connects to the Internet through interface GigabitEthernet0/0. DHCP server running on GNS3 NAT assigns the IP address for this interface.

```
ISP1(config)# interface GigabitEthernet 0/0
ISP1(config-if)# description Link to Simulated Internet
ISP1(config-if)# ip address dhcp
ISP1(config-if)# ip nat outside
ISP1(config-if)# no shutdown
```

```
ISP1(config)# interface GigabitEthernet 1/0
ISP1(config-if)# description Link to Company Inc.
ISP1(config-if)# ip address 198.10.10.1 255.255.255.252
ISP1(config-if)# ip nat inside
ISP1(config-if)# no shutdown
```

b) eBGP Configuration

```
ISP1(config)# ip prefix-list static_default permit 0.0.0.0/0
ISP1(config)# route-map static_default permit 10
ISP1(config-route-map)# match ip address prefix-list static_default

ISP1(config)# router bgp 64501
ISP1(config-router)# neighbor 198.10.10.2 remote-as 64500
ISP1(config-router)# neighbor 198.10.10.2 ttl-security hops 1
ISP1(config-router)# neighbor 198.10.10.2 password isp1md5pass
ISP1(config-router)# neighbor 198.10.10.2 route-map static_default out
ISP1(config-router)# network 0.0.0.0 mask 0.0.0.0
```

c) NAT Configuration

To have a real internet connection, translate the subnets 195.1.1.0/24 and 198.10.10.0/30 to the IP address that is received for interface GigabitEthernet0/0.

```
ISP1(config)# ip access-list standard 1
ISP1(config-std-nacl)# permit 195.1.1.0 0.0.0.255
ISP1(config-std-nacl)# permit 198.10.10.0 0.0.0.3
ISP1(config-std-nacl)# deny any
ISP1(config)# ip nat inside source list 1 interface GigabitEthernet 0/0 overload
```

d) DNS Configuration

```
ISP1(config)# ip domain-lookup
ISP1(config)# ip name-server 8.8.8.8 8.8.4.4
```

e) Configuration file

The `/service_provider/` directory located in the https://github.com/angelavarcila/emulatedNet_configFiles.git repository stores the configuration files for the ISP1 and ISP2 devices for consultation purposes.

A.6 Credentials summary

Table A.3 shows the list of the credentials for all devices in the emulated network.

Table A.3 Credentials summary for emulated network elements.

Device		Login/User	Password
Cisco and Arista Network Elements	Local User - Level 1	admin	cisco
	Local User - Level 15	admin15	cisco15
	Local enable		cisco

	Radius User - Level 1	raadmin	racisco
	Radius User - Level 15	raadmin15	racisco15
	Radius enable		racisco
Juniper Firewall	Local User	root	Juniper
PCs		tc	tc
ISPs		not configured for authentication	
ServerA	Operative sytem	server1	server1
ServerB	Operative system	appliance	zabbix

Annex B: Description of raw data

Table B.1 Parameters collected from nProbe flow files.

No.	PARAMETER	DESCRIPTION
1	IN_BYTES	Incoming flow bytes.
2	IN_PKTS	Incoming flow packets.
3	PROTOCOL	The protocol field of IP header (8 bits).
4	PROTOCOL_MAP	IP protocol name.
5	SRC_TOS	Type of Service byte setting when entering incoming interface.
6	TCP_FLAGS	Cumulative of all the TCP flags seen for the flow.
7	L4_SRC_PORT	TCP/UDP source port number.
8	L4_SRC_PORT_MAP	Layer 4 source port symbolic name.
9	IPV4_SRC_ADDR	IP version 4 source address.
10	IPV4_SRC_MASK	IP version 4 source subnet mask.
11	INPUT_SNMP	Index of SNMP input interface.
12	L4_DST_PORT	TCP/UDP destination port number i.e.: FTP, Telnet, or equivalent.
13	L4_DST_PORT_MAP	Layer 4 destination port symbolic name.
14	L4_SRV_PORT	Layer 4 server port.
15	L4_SRV_PORT_MAP	Layer 4 server port symbolic name.
16	IPV4_DST_ADDR	IP version 4 destination address.
17	IPV4_DST_MASK	IP version 4 dest subnet mask.
18	OUTPUT_SNMP	Index of SNMP output interface.
19	IPV4_NEXT_HOP	IP version 4 address of next-hop router.
20	SRC_AS	Source BGP autonomous system number.
21	DST_AS	Destination BGP autonomous system number.
22	LAST_SWITCHED	System uptime at which the last packet of the flow was switched.
23	FIRST_SWITCHED	System uptime at which the first packet of the flow was switched.
24	OUT_BYTES	Outgoing flow bytes.
25	OUT_PKTS	Outgoing flow packets.
26	IPV6_SRC_ADDR	IP version 6 Source Address.
27	IPV6_DST_ADDR	IP version 6 Destination Address.
28	IPV6_SRC_MASK	Length of the IP version 6 source mask in contiguous bits.
29	IPV6_DST_MASK	Length of the IP version 6 destination mask in contiguous bits.

30	ICMP_TYPE	Internet Control Message Protocol (ICMP) packet type; reported as ((ICMP Type*256) + ICMP code).
31	SAMPLING_INTERVAL	The rate at which packets are sampled.
32	SAMPLING_ALGORITHM	The type of algorithm used for sampled NetFlow
33	FLOW_ACTIVE_TIMEOUT	Timeout value (in seconds) for active flow entries in the NetFlow cache.
34	FLOW_INACTIVE_TIMEOUT	Timeout value (in seconds) for inactive flow entries in the NetFlow cache.
35	ENGINE_TYPE	Type of flow switching engine: RP = 0, VIP/Linecard = 1
36	ENGINE_ID	ID number of the flow switching engine.
37	TOTAL_BYTES_EXP	Total number of bytes exported.
38	TOTAL_PKTS_EXP	Total number of flow packets exported.
39	TOTAL_FLOWS_EXP	Total exported flows.
40	MIN_TTL	Minimum TTL on incoming packets of the flow.
41	MAX_TTL	Maximum TTL on incoming packets of the flow.
42	DST_TOS	Type of Service byte setting when exiting outgoing interface.
43	IN_SRC_MAC	Source MAC Address.
44	SRC_VLAN	Virtual LAN identifier associated with ingress interface.
45	DST_VLAN	Virtual LAN identifier associated with egress interface.
46	IP_PROTOCOL_VERSION	[4=IPv4],[6=IPv6]
47	DIRECTION	If 0, it is incoming flow. If 1, it is an outgoing flow.
48	IPV6_NEXT_HOP	IPv6 address of the next-hop router.
49	OUT_DST_MAC	Destination MAC Address.
50	APPLICATION_ID	Cisco NBAR Application Id.
51	PACKET_SECTION_OFFSET	Packet section offset.
52	SAMPLED_PACKET_SIZE	Sampled packet size.
53	SAMPLED_PACKET_ID	Sampled packet id.
54	EXPORTER_IPV4_ADDRESS	Exporter IP version 4 Address.
55	EXPORTER_IPV6_ADDRESS	Exporter IP version 6 Address.
56	FLOW_ID	Serial Flow Identifier.
57	FLOW_START_SEC	Seconds of the first flow packet.
58	FLOW_END_SEC	Seconds of the last flow packet.
59	FLOW_START_MILLISECONDS	Milliseconds of the first flow packet.
60	FLOW_END_MILLISECONDS	Milliseconds of the last flow packet.
61	BIFLOW_DIRECTION	1=initiator, 2=reverseInitiator
62	OBSERVATION_POINT_TYPE	Observation point type.
63	OBSERVATION_POINT_ID	Observation point id.
64	SELECTOR_ID	Selector id.
65	IPFIX_SAMPLING_ALGORITHM	Sampling algorithm.
66	SAMPLING_SIZE	Number of packets to sample.
67	SAMPLING_POPULATION	Sampling population.
68	FRAME_LENGTH	Original L2 frame length.

69	PACKETS_OBSERVED	Total number of packets seen.
70	PACKETS_SELECTED	Number of packets selected for sampling.
71	SELECTOR_NAME	Sampler name.
72	APPLICATION_NAME	Name associated with a classification.
73	FRAGMENTS	Number of fragmented flow packets.
74	APPL_LATENCY_MS	Application latency (msec)
75	NUM_PKTS_UP_TO_128_BYTES	Number of packets with size S, where $S \leq 128$
76	NUM_PKTS_128_TO_256_BYTES	Number of packets with size S, where $128 < S \leq 256$
77	NUM_PKTS_256_TO_512_BYTES	Number of packets with size S, where $256 < S < 512$
78	NUM_PKTS_512_TO_1024_BYTES	Number of packets with size S, where $512 < S < 1024$
79	NUM_PKTS_1024_TO_1514_BYTES	Number of packets with size S, where $1024 < S \leq 1514$
80	NUM_PKTS_OVER_1514_BYTES	Number of packets with size S, where $S > 1514$
81	CUMULATIVE_ICMP_TYPE	Cumulative OR of ICMP type packets
82	SRC_IP_COUNTRY	Country where the src IP is located
83	SRC_IP_CITY	City where the src IP is located
84	DST_IP_COUNTRY	Country where the dst IP is located
85	DST_IP_CITY	City where the dst IP is located
86	FLOW_PROTO_PORT	L7 port that identifies the flow protocol or 0 if unknown.
87	UPSTREAM_TUNNEL_ID	Upstream tunnel identifier (e.g. GTP TEID) or 0 if unknown.
88	LONGEST_FLOW_PKT	Longest packet of the flow
89	SHORTEST_FLOW_PKT	Shortest packet of the flow
90	RETRANSMITTED_IN_PKTS	Number of incoming retransmitted TCP flow packets
91	RETRANSMITTED_OUT_PKTS	Number of outgoing retransmitted TCP flow packets
92	OOORDER_IN_PKTS	Number of out-of-order TCP incoming flow packets
93	OOORDER_OUT_PKTS	Number of out-of-order TCP outgoing flow packets
94	UNTUNNELED_PROTOCOL	Untunneled IP protocol byte
95	UNTUNNELED_IPV4_SRC_ADDR	Untunneled IP version 4 source address
96	UNTUNNELED_L4_SRC_PORT	Untunneled IP version 4 source port
97	UNTUNNELED_IPV4_DST_ADDR	Untunneled IP version 4 destination address
98	UNTUNNELED_L4_DST_PORT	Untunneled IP version 4 destination port
99	L7_PROTO	Layer 7 protocol (numeric)
100	L7_PROTO_NAME	Layer 7 protocol name
101	DOWNSTREAM_TUNNEL_ID	Downstream tunnel identifier (e.g. GTP TEID) or 0 if unknown
102	FLOW_USER_NAME	Flow username of the tunnel (if known)
103	FLOW_SERVER_NAME	Flow server name (if known)
104	PLUGIN_NAME	Plugin name used by this flow (if any)
105	NUM_PKTS_TTL_EQ_1	Number of packets with $TTL = 1$
106	NUM_PKTS_TTL_2_5	Number of packets with $1 < TTL \leq 5$
107	NUM_PKTS_TTL_5_32	Number of packets with $5 < TTL \leq 32$
108	NUM_PKTS_TTL_32_64	Number of packets with $32 < TTL \leq 64$
109	NUM_PKTS_TTL_64_96	Number of packets with $64 < TTL \leq 96$

110	NUM_PKTS_TTL_96_128	Number of packets with 96 < TTL <= 128
111	NUM_PKTS_TTL_128_160	Number of packets with 128 < TTL <= 160
112	NUM_PKTS_TTL_160_192	Number of packets with 160 < TTL <= 192
113	NUM_PKTS_TTL_192_224	Number of packets with 192 < TTL <= 224
114	NUM_PKTS_TTL_224_255	Number of packets with 224 < TTL <= 255
115	IN_SRC_OSI_SAP	OSI Source SAP (OSI Traffic Only)
116	OUT_DST_OSI_SAP	OSI Destination SAP (OSI Traffic Only)

Table B.2 Parameters collected from Rsyslog data base.

No.	PARAMETER	FORMAT	DESCRIPTION		
1	Date	date	The system time when the syslog message was generated.		
2	Facility	text	<div><div><div>Numerical Code</div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div><div>8</div><div>9</div><div>10</div><div>11</div><div>12</div><div>13</div><div>14</div><div>15</div><div>16 - 23</div></div><div><div>kern</div><div>user</div><div>mail</div><div>daemon</div><div>auth</div><div>syslog</div><div>lpr</div><div>news</div><div>uucp</div><div>cron</div><div>security</div><div>ftp</div><div>ntp</div><div>logaudit</div><div>logalert</div><div>clock</div><div>local0 -7</div></div><div><div>Kernel messages.</div><div>User-level messages.</div><div>Mail system.</div><div>System daemons.</div><div>Security/authorization messages.</div><div>Messages generated internally by syslogd.</div><div>Line printer subsystem.</div><div>Network news subsystem.</div><div>UUCP subsystem.</div><div>Clock daemon.</div><div>Security/authorization messages.</div><div>FTP daemon.</div><div>NTP subsystem.</div><div>Log audit.</div><div>Log alert.</div><div>Clock daemon (note 2).</div><div>local use 0 (local0) - local use 7 (local7).</div></div></div>	Facility	Description
3	Severity	text	<div><div><div>Numerical Code</div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>7</div></div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div>Emergency</div><div>Alert</div><div>Critical</div><div>Error</div><div>Warning</div><div>Notice</div><div>Informational</div><div>Debug</div></div></div><div><div>The system is unusable.</div><div>An action must be taken immediately.</div><div>Critical conditions.</div><div>Error conditions.</div><div>Warning conditions.</div><div>Normal but significant condition.</div><div>Informational messages.</div><div>Debug-level messages.</div></div></div>	Severity	Description
4	Host	text	The administratively-assigned name for the device that originally sent the syslog message.		
5	Syslogtag	text	A string prefix to the Syslog entry which usually indicates the name of the program or process that generated the message.		
6	ProcessID	text	Identification of the originator process.		
7	MessageType	text	The type of message.		
8	Message	text	Free-form text that provides the message with information about the event.		

Table B.3 Parameters collected from Zabbix data base.

No.	TYPE	PARAMETER	FORMAT	DESCRIPTION
1	Traffic descriptors	Bits_received	numeric	The total number of bits received on each interface of the device.
2		Bits_sent	numeric	The total number of bits sent by each interface of the device.
3		Inbound_packets_discarded	numeric	The total number of inbound packets discarded by each interface of the device.
4		Inbound_packets_with_errors	numeric	The total number of packets that contained errors, received on each interface of the device.
5		Interface_type	numeric	The type of each interface of the device. The value is based on the IANAifType defined by Internet Assigned Numbers Authority (IANA).
6		Operational_status	numeric	The current operational state of each interface of the device.
7		Outbound_packets_discarded	numeric	The total number of packets discarded out of each interface of the device.
8		Outbound_packets_with_errors	numeric	The total number of packets that could not be transmitted by each interface of the device because of errors.
9		Speed	numeric	Current bandwidth for each interface of the device.
10	Status descriptors	Device_uptime	numeric	The time since the device was last re-initialized.
11		SNMP_availability	numeric	0 - if the device has not SNMP enabled. 1 - if the device has SNMP enabled.
12		ICMP_response_time	numeric	The time between the echo request and echo response messages from the SNMP server to the device. If the host is not available (timeout reached), the item will return 0.
13		ICMP_loss	numeric	Percentage of lost packets.
14		ICMP_ping	numeric	Device accessibility by ICMP ping. 0 - ICMP ping fails. 1 - ICMP ping successful.
15	Identification descriptors	Device_name	text	The administratively-assigned name for the device.
16		System_object_ID	text	The vendor's authoritative identification of the device.
17		Device_location	text	The physical location of the device (e.g., "building A, floor 3, RAC 2").
18		Device_description	text	The device description. This value includes the name of the manufacturer, the full name and version of the software operating-system, and networking software.
19		Device_contact_details	text	The administrative contact details of the device (e.g., e-mail of the responsible for the device).

Annex C: Extraction application for Zabbix

The software application for SOFI dataset extraction (SOFIApp) is an open-source java project found in <https://github.com/angelavarcila/zabbixDataSets> based on the Zabbix database version 3.4.2. SOFIApp is made up of four independent modules that run specific tasks. The processes must be executed in the order shown in Figure C.1 to build a SOFI dataset for a peripheral device.

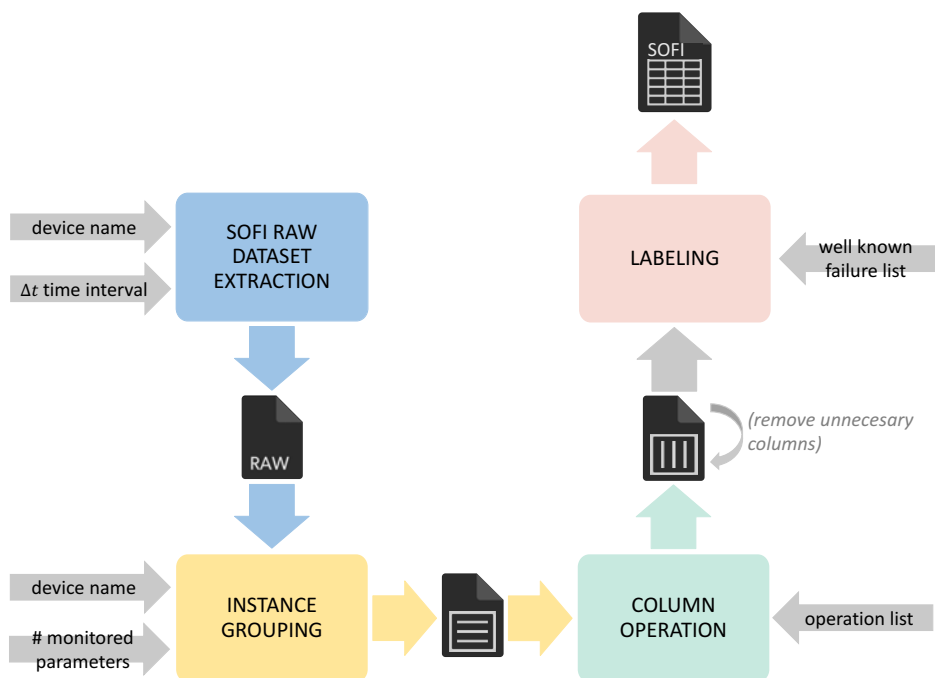


Figure C.1 Modules of SOFIApp extraction application.

The first step is to execute the module or process "SOFI raw dataset extraction" that is responsible for extracting the raw data of the monitorization of a particular peripheral device. The second step corresponds to the "instance grouping" process that takes a SOFI raw dataset and groups the instances resulting from the same polling. The third step is grouping the parameters from the same interface level. For this purpose, the process "column operation" allows operating a column set applying any defined formula in section 2.1.2 of Chapter IV. So, this process creates as many new columns that contain the results as performed operations.

Before doing the labeling process, the columns that were operands in any of the operations must be eliminated, because they become unnecessary. Any desired tool that allows removing columns can perform this step. Once the removal is done, the result is the unlabeled SOFI dataset.

Finally, the following step is to execute the "labeling" process, which generates a class column for the SOFI dataset with the labels according to a list of well-known failures.

These modules are described in more detail below.

C.1 SOFI raw dataset extraction process

This process receives two inputs, the name of the peripheral device for which being built the SOFI dataset and the time interval that will cover this dataset.

As shown in Figure C.2, the first function of this module requests the Zabbix database the monitored parameter list in the specified peripheral device. A second task obtains all the timestamps within the specified time interval, in which there is a record of any monitored parameter. The last function creates an instance for each timestamp with the monitored parameters at that moment. If a parameter does not have a record for a timestamp, its value is set to -1.

In this way, the SOFI raw dataset of size $q \times r$ is obtained, where $r - 1$ is the number of monitored parameters and q the number of timestamps with some monitoring record.

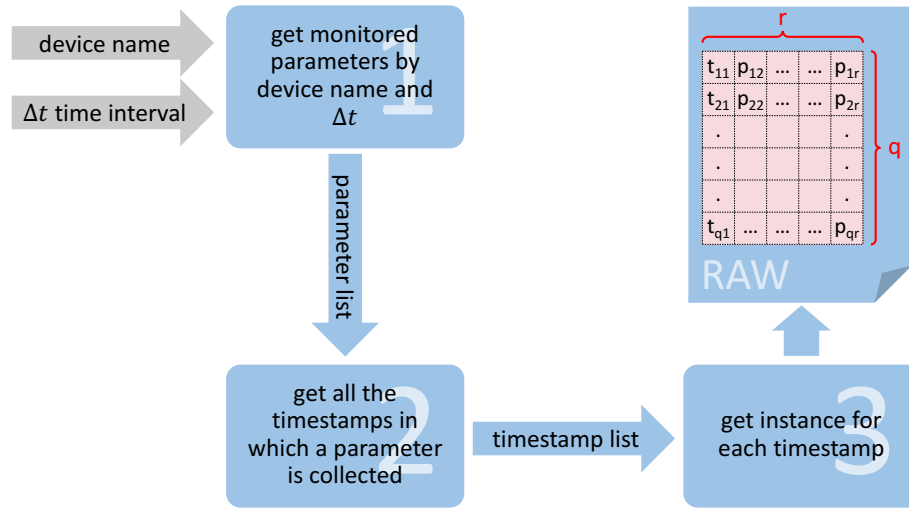


Figure C.2 SOFI raw dataset extraction process.

C.2 Instance grouping process

This process receives three inputs from the user, the SOFI raw dataset ($q \times r$ size) in which grouping will be performed, the device name to which the dataset belongs, and the number of monitored parameters in the device.

As shown in Figure C.3, the first function of this module identifies, in the raw dataset, the instance groups from the same polling. This function represents each group by a time interval and a timestamp. The timestamp is the time in which the third quartile of the data collected in the said

interval is located. With these two values found, the second function obtains a single instance for each interval, being the third quartile timestamp the time assigned to the instance.

In this way, the dimensions of the new dataset (grouped SOFI raw dataset) are $m \times r$, where $m < q$, and $m = \text{number of pollings}$.

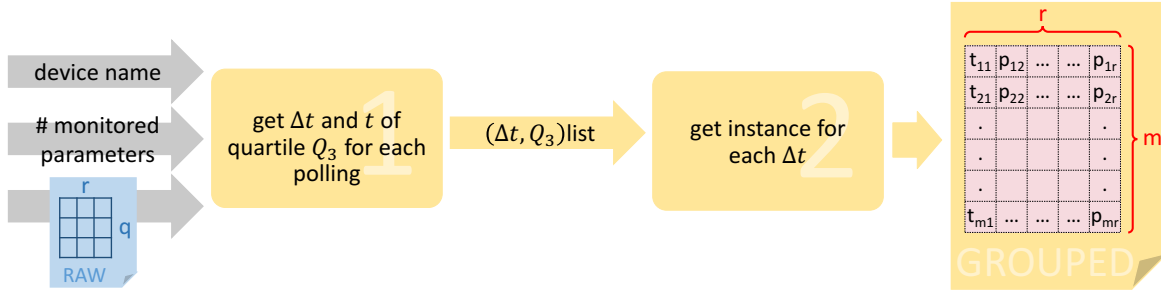


Figure C.3 Instance grouping process.

C.3 Column operation process

This process receives two files as input, the dataset on which the operations will be performed ($m \times r$ size), and a file that details the desired operations. Figure C.4 specifies the format of the operations file with an example.

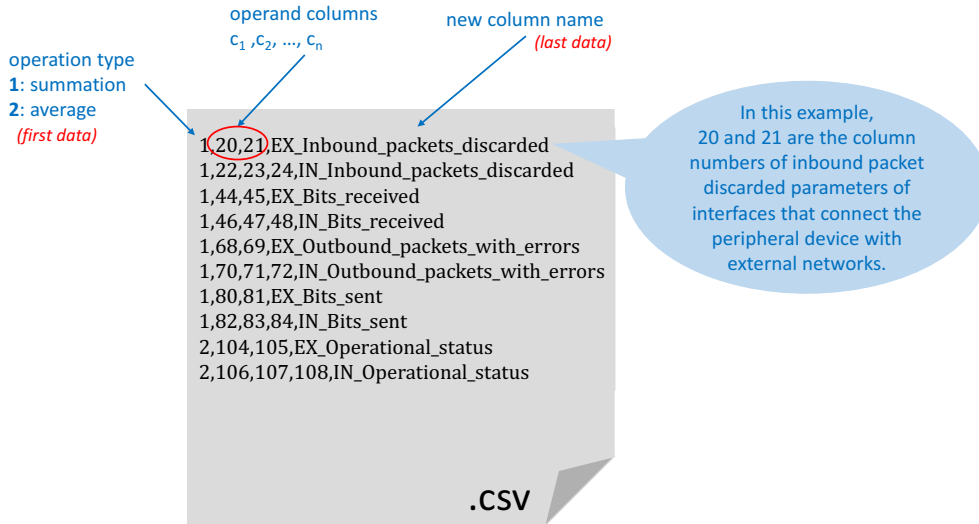


Figure C.4 Format of the operations list file.

As shown in Figure C.5, this module executes a single function that reads the operation list and creates a new column for each operation with the resulting values. In this way, a new dataset (grouped and operated SOFI raw dataset) of dimension $m \times p$ is obtained, where $p = r + \text{number of operations}$.



Figure C.5 Column operation process.

Remove all the columns which were operands before executing the fourth module. Thus, a dataset of dimension $m \times n$ with a fixed number of features is obtained. This new dataset corresponds to the unlabeled SOFI dataset.

C.4 Labeling process

This process receives two files as input, the unlabeled SOFI dataset ($m \times n$ size), and a file that details the well-known failures. Figure C.6 specifies the format of the failures file.

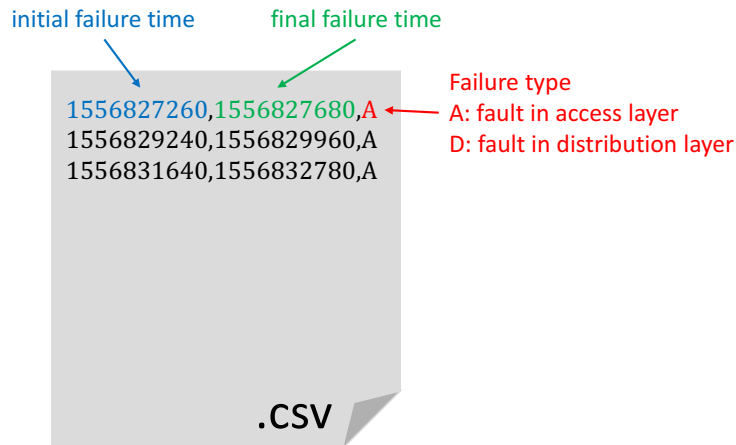


Figure C.6 Format of the failure list file.

As shown in Figure C.7, this module executes a single function that creates the class column and assigns the corresponding label (A for failures in the access layer, D for failure in the distribution layer, and NE if there are not failure) according to the instance timestamp and failure time interval. The result is the labeled SOFI dataset.

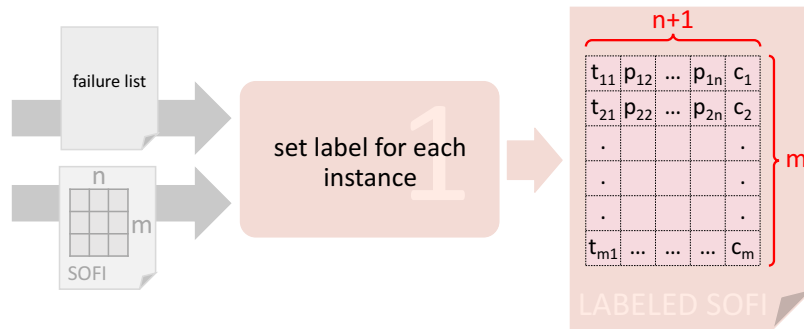


Figure C.7 SOFI labeling process.

C.5 Other functions

In addition to the SOFI building process, SofiApp allows to perform other four tasks:

- Get all events triggered by a specific network device within a specified time interval.
- Get the history of all the monitored parameters in a particular network device within a specified time interval.
- Get from the distribution of monitored parameters, the timestamps of events triggered for those parameters (third quartile).
- Generate the SOFI dataset class column from a set of events.

C.6 Zabbix database

Zabbix has an extensive database of 140 tables. Nevertheless, SOFIApp development involves only the tables shown in Figure C.8. The description of each table is in **¡Error! No se encuentra el origen de la referencia..**

Table C.1 Description of the tables from the Zabbix database used by SOFIApp.

TABLE TYPE	TABLE NAME	DESCRIPTION
Device description	hosts	It stores all monitored devices by Zabbix.
	interface	It stores the interfaces for SNMP communication with each host.
Monitorization configuration	items	It stores the monitored parameters of each host (such as the number of bits sent on the Gi0/0 interface of Switch A device, the name of the Router A device, etc.).
	functions	It relates each item with triggers that can launch an event for the item.
	triggers	It stores all triggers that can activate an event during monitoring.
History	history	They contain the monitoring history according to the type of item.
	history_log	
	history_text	
	history_uint	
	history_str	

Events	event	It stores all events that occur on monitored devices and interfaces.
	event_recovery	It relates two events to know which events mean the recovery of another event.

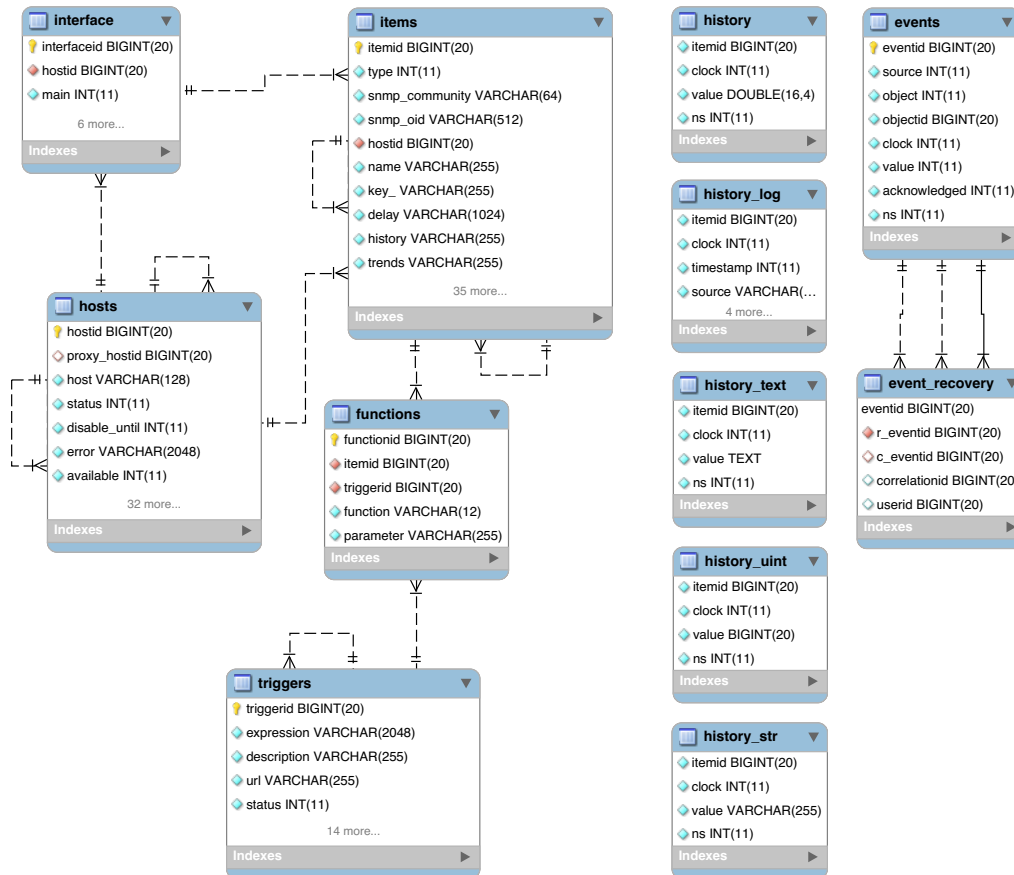


Figure C.8 Tables from the Zabbix database used by SOFIApp.

Annex D: Experiments charts

This annex contains all the results obtained when executing the PALADIN model's diagnosis module using 25 different algorithms for data stream mining and the two datasets built from monitoring the monitored network's two peripheral devices.

Each section of the annex sorts the results of a performance parameter. Section D.1 presents the prequential AUC charts, section D.2 contains the G-mean charts, section D.3 shows the Cohen's Kappa Coefficient charts, section D.4 contains the MCC charts, and section D.5 presents the Recall charts.

All results are organized in tables. The first column of the tables indicates the algorithm used. The second column contains the graphs obtained for the corresponding parameter using the first peripheral device's data set. The third column contains the graphs for the second peripheral device.

As the Figure D.1 shows, each graph represents tested algorithms' performance curves according to the corresponding metric. The vertical axis indicates the metric values, and the horizontal axis indicates the number of incoming instances that have arrived until the measurement time. The black circles series are the results with the diagnosis module approach, that is, with the rebalance stream strategy. The red squares series are the results of the online base algorithm performance without rebalancing or concept-drift treatment.

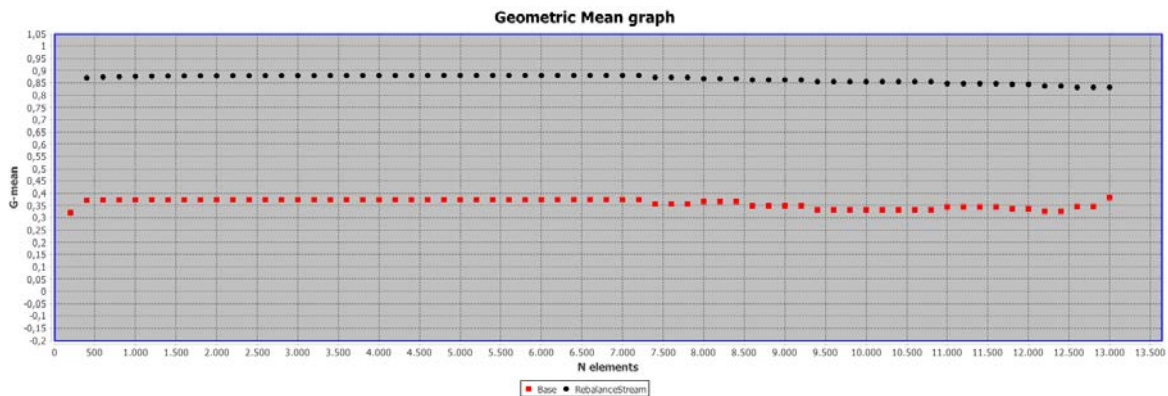
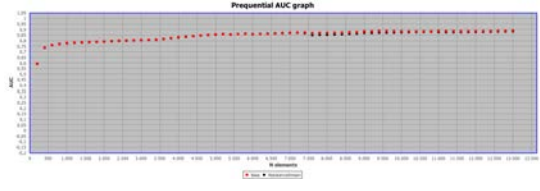
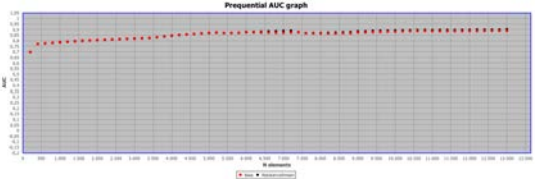
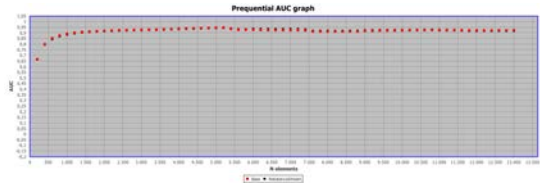
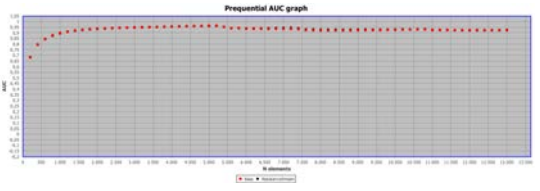
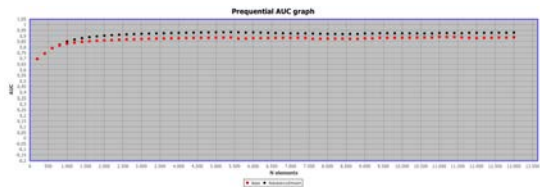
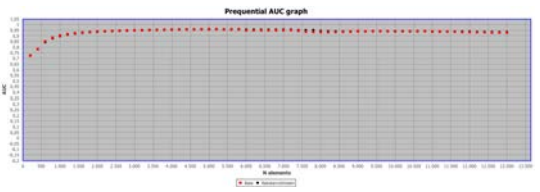
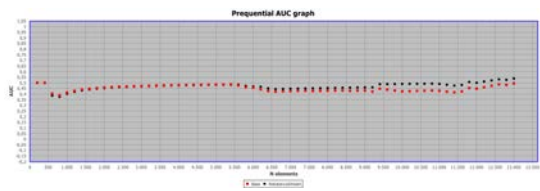
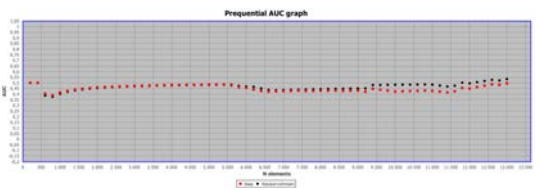
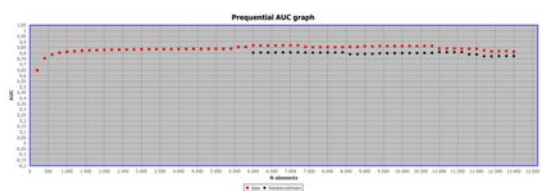
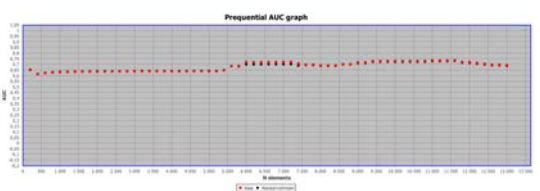
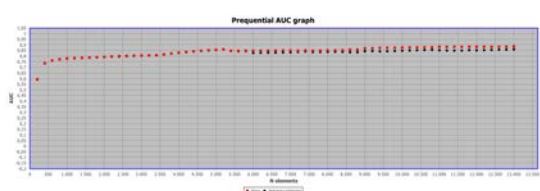
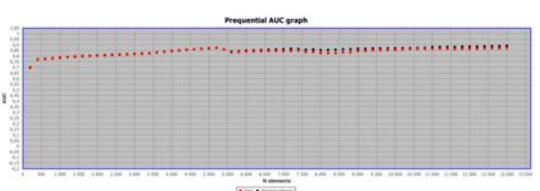
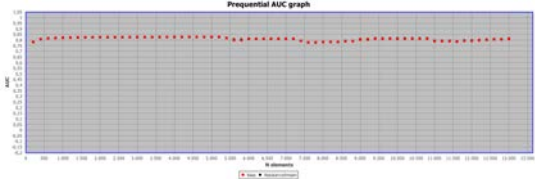
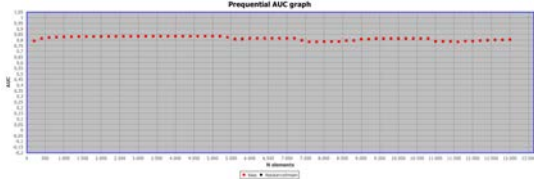
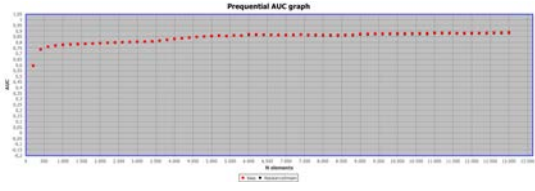
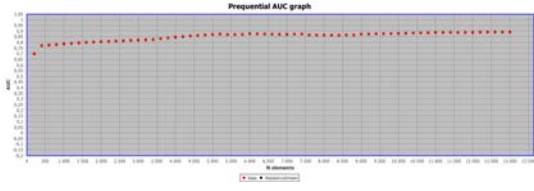
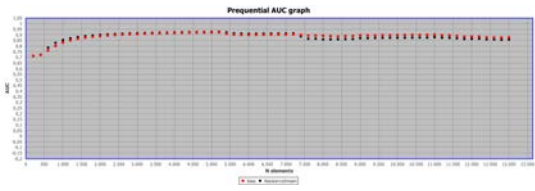
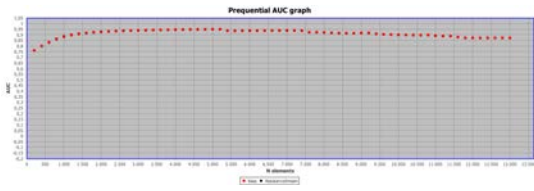
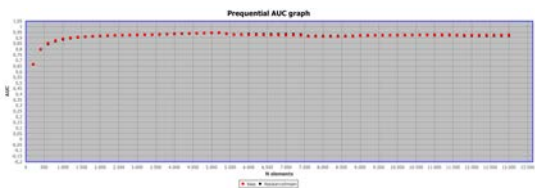
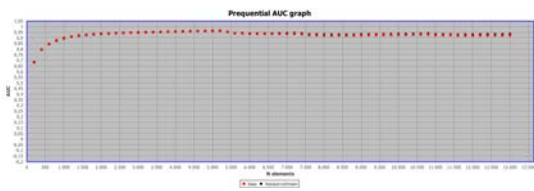
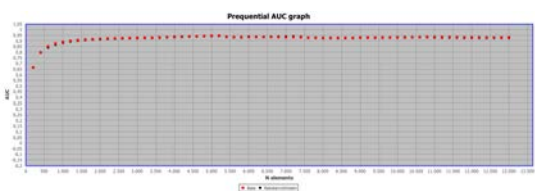
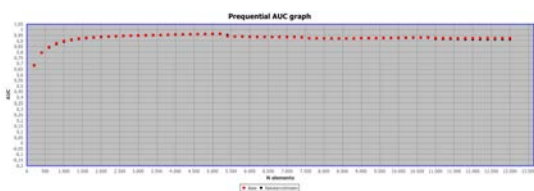
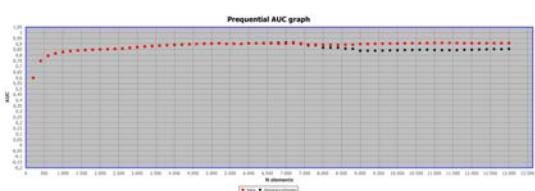
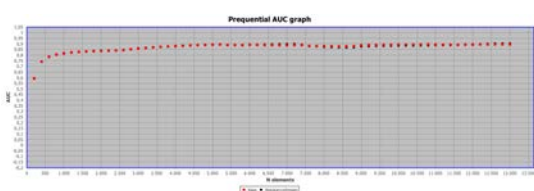
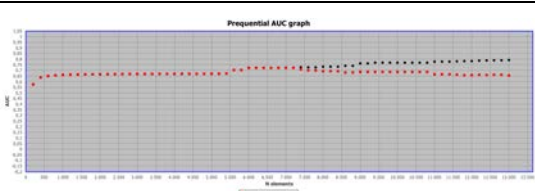
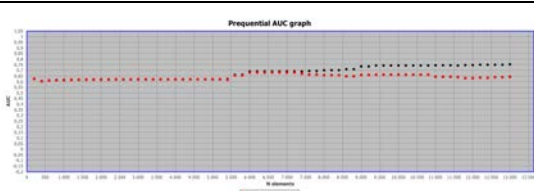
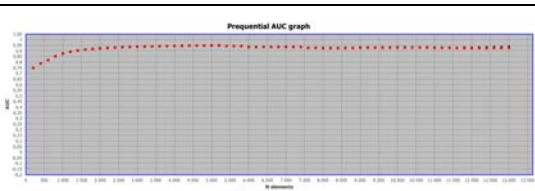
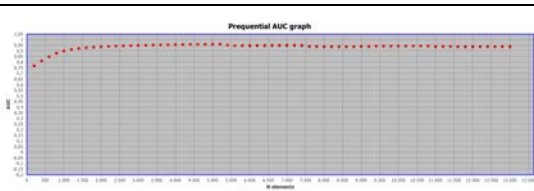


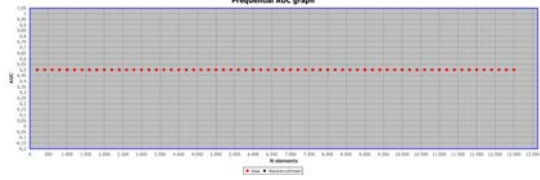
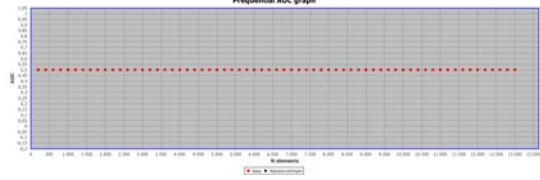
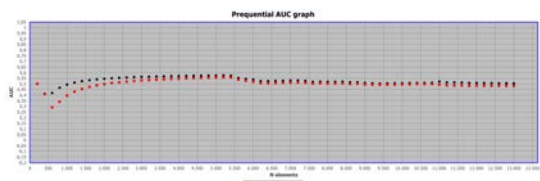
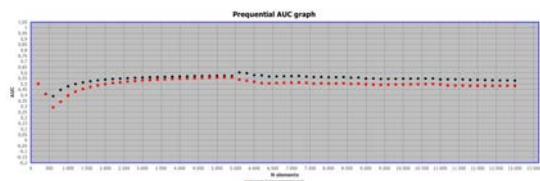
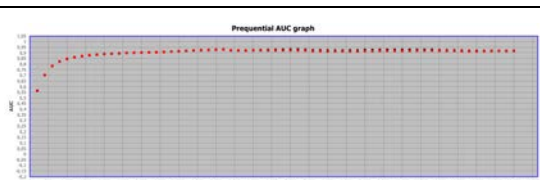
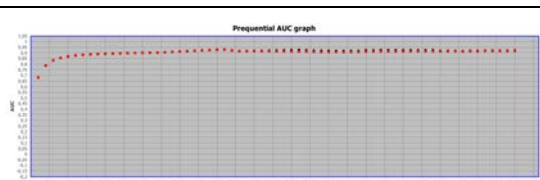
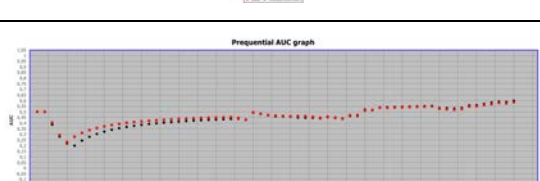
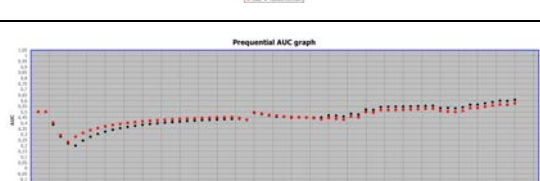
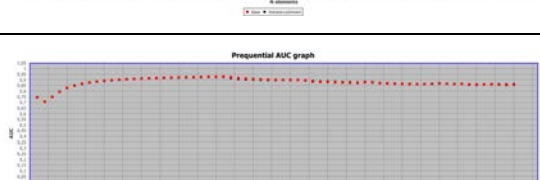
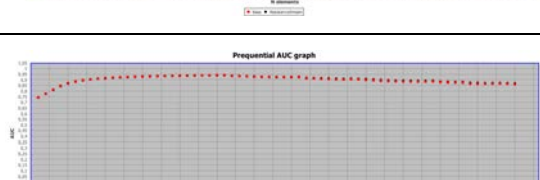
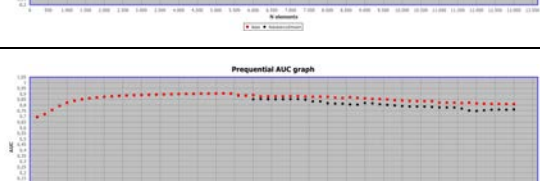
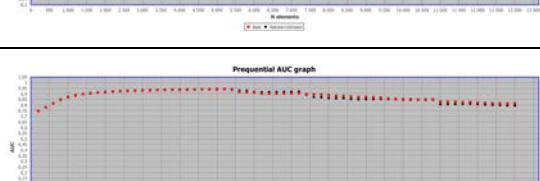
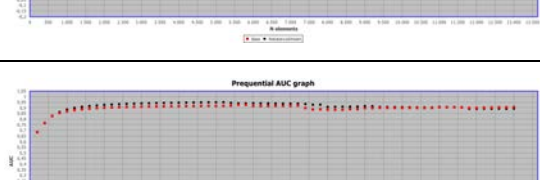
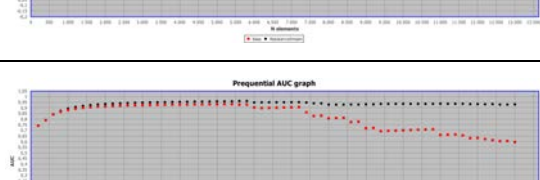
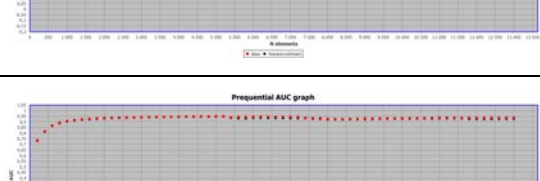
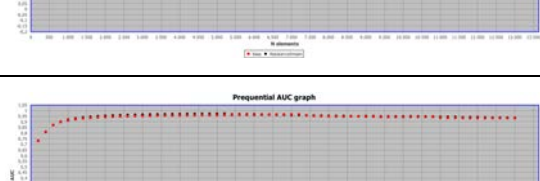




Figure D.1 Performance curves format.

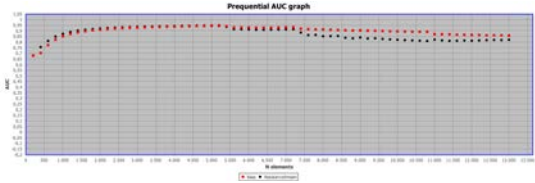
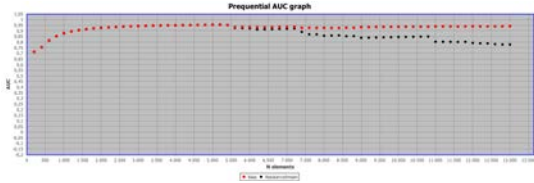
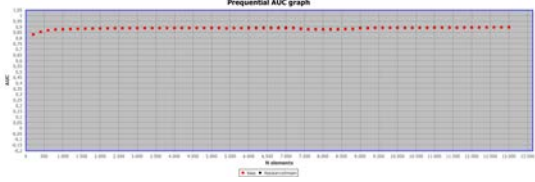
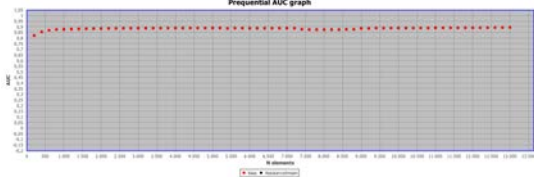
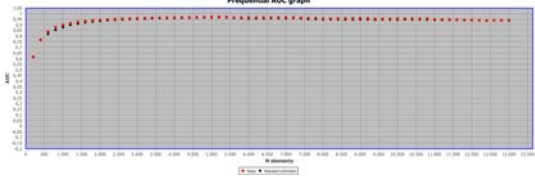
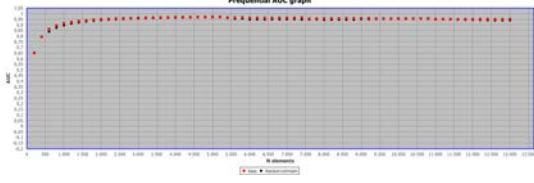
D.1 Prequential AUC charts

Table D.1 Charts of results of the Prequential AUC performance metric.

Algorithm	SwitchCore-I Dataset	SwitchCore-II Dataset
1		
CVFDT		
2		
Online Bagging		
3		
Online Boosting		
4		
Accuracy Weighted Ensemble (AWE)		
5		
Dynamic Weighted Majority (DWM)		
6		
Hoeffding Option Tree		

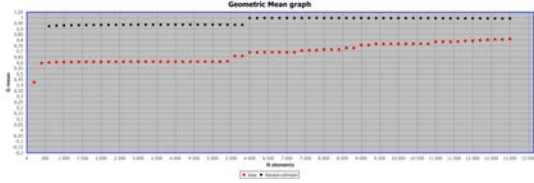
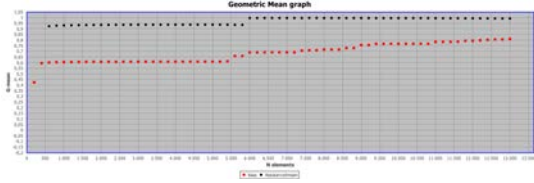
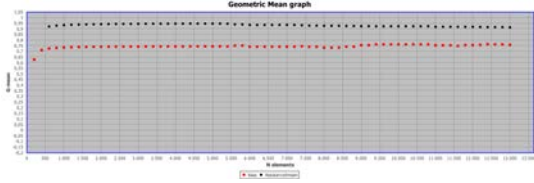
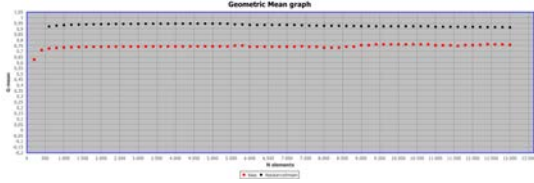
7	PEGASOS (Primal Estimated sub-Gradient Solver for SVM)		
8	CVFDT _{HC}		
9	Paired learning		
10	ADWIN Bagging		
11	ASHT (Adaptive-Size Hoeffding Tree) Bagging		
12	Hoeffding Adaptive Tree		
13	Online Coordinate Boosting		
14	Leveraging Bagging		


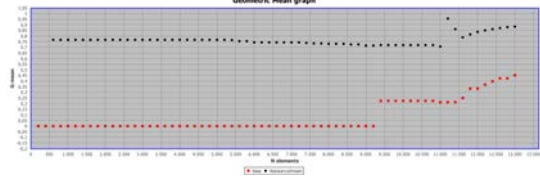
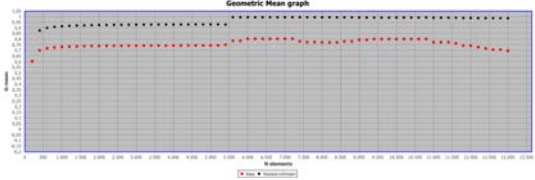
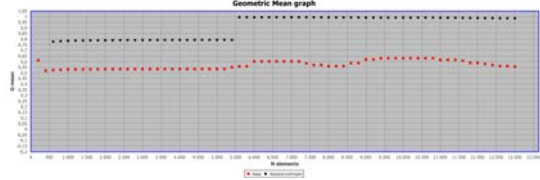
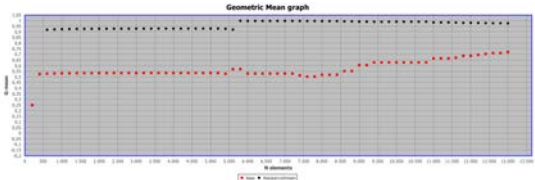
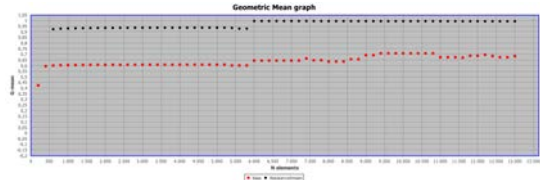
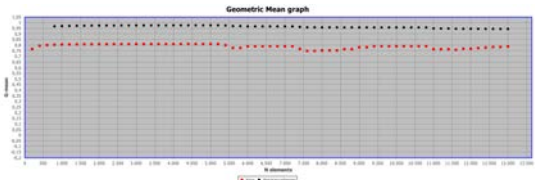
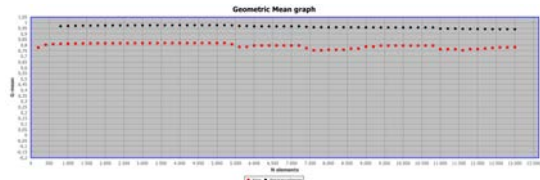
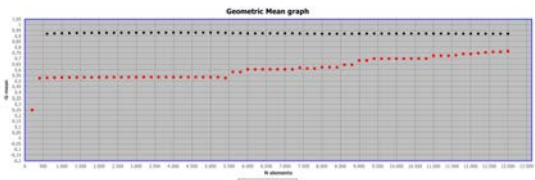
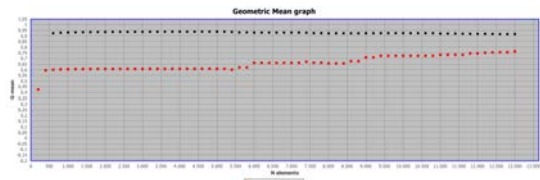
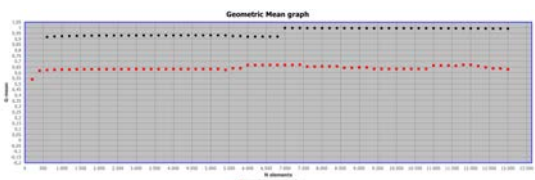
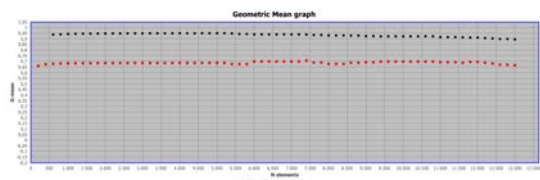
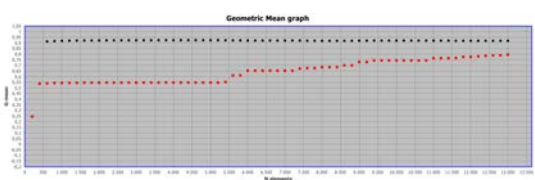
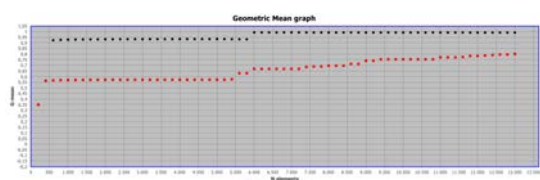
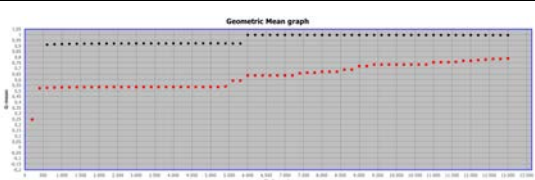
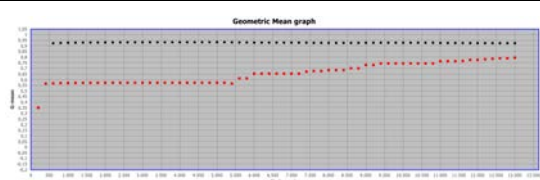
15	ADAGRAD		
16			
17	Learn++-NSE		
18			
19	OSBoost (Online Smooth Boost)		
20			
21	Accuracy Updated Ensemble (AUE2)		
22			
23	Dynamic Adaptation to Concept Changes (DACC)		
24			
25	Recurring Concept Drifts (RCD)		
26			
27	ADOB (Adaptable Diversity-based Online Boosting)		
28			
29	Boosting-like Online Learning Ensemble (BOLE)		
30			

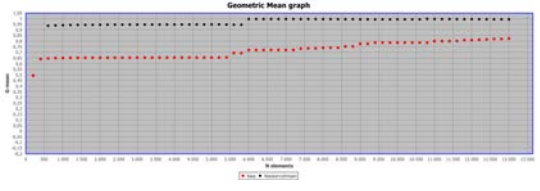
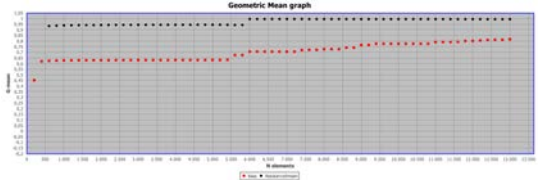
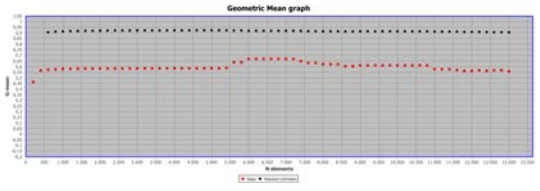
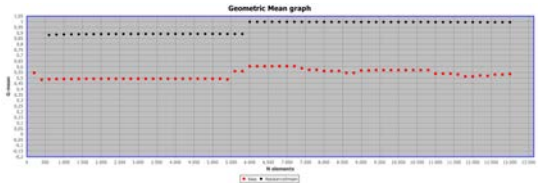
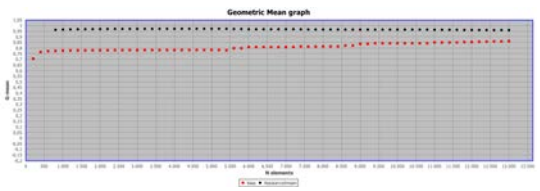
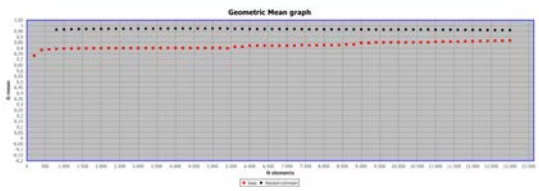
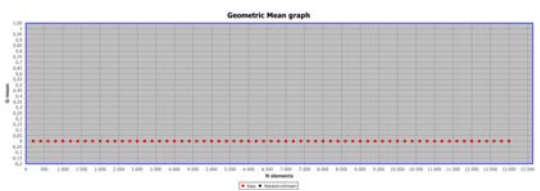
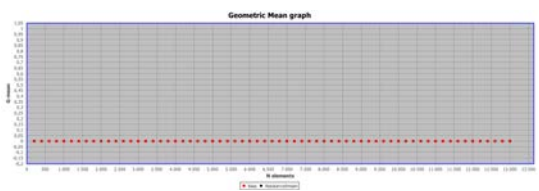
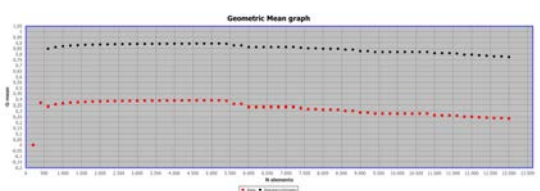
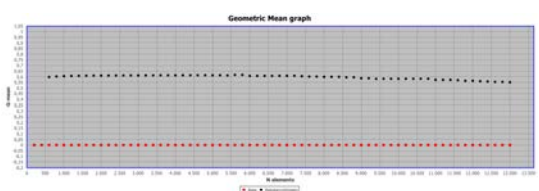
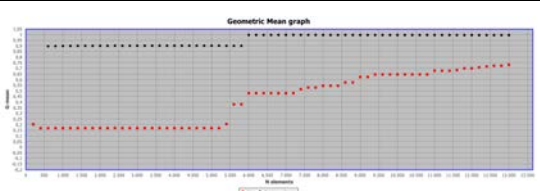
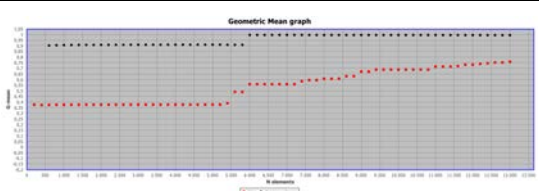
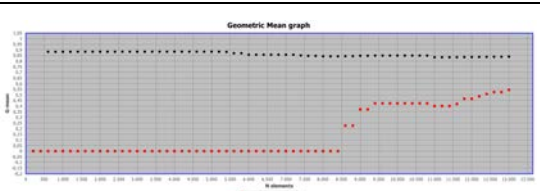
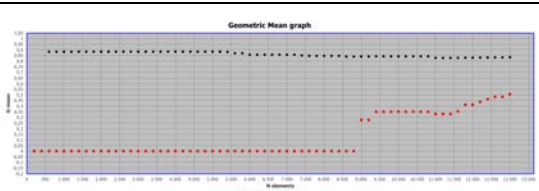
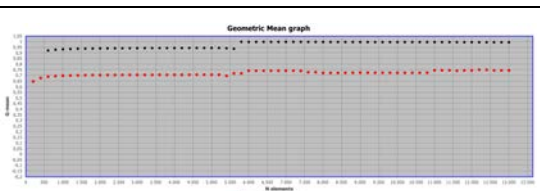
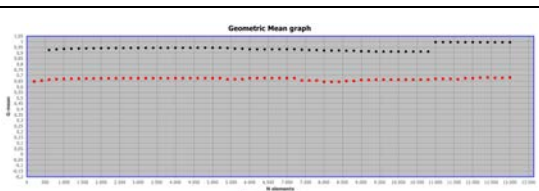




23	Hoeffding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)		
24			
25	Adaptive Random Forest (ARF)		

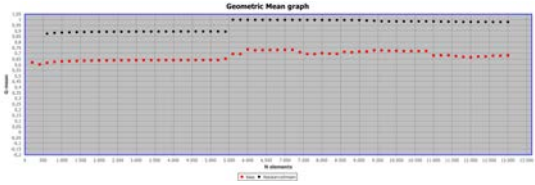
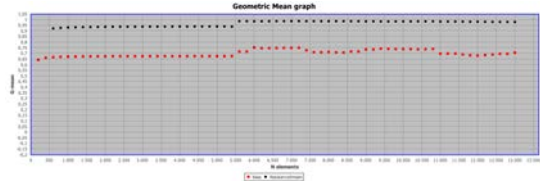
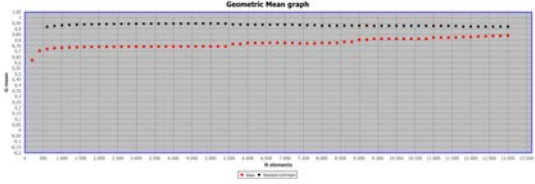
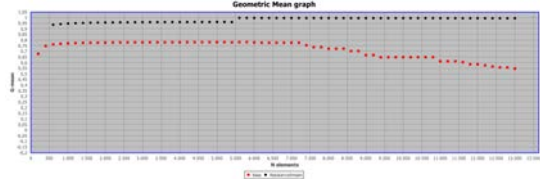
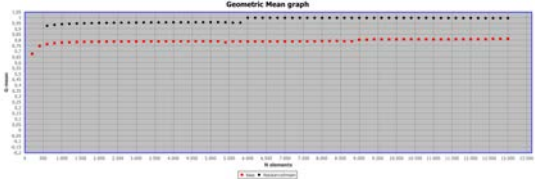
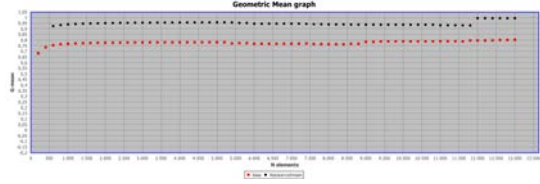
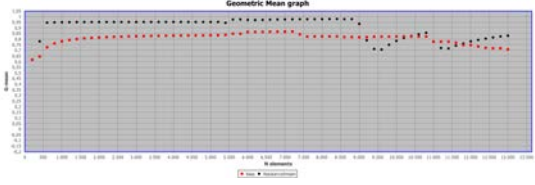
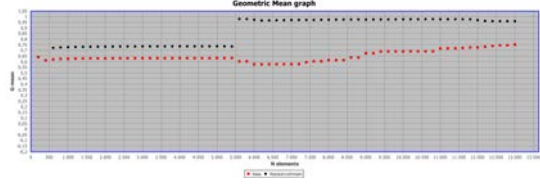
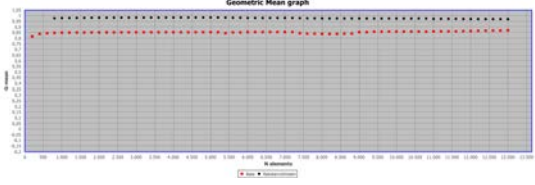
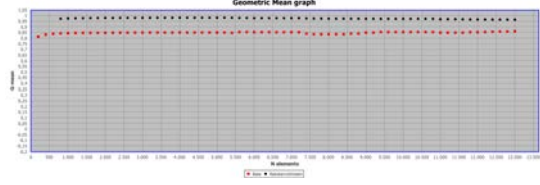
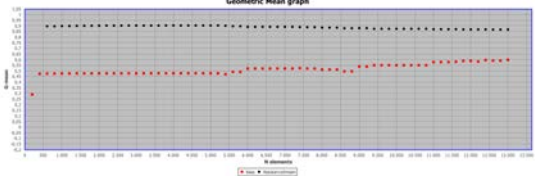
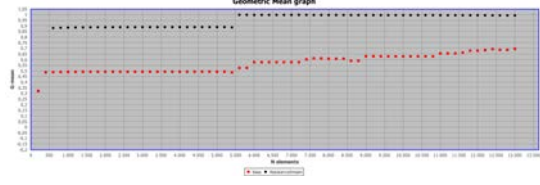
D.2 Geometric mean (G-Mean) charts

Table D.2 Charts of results of the prequential GMean performance metric.

Algorithm	SwitchCore-I Dataset	SwitchCore-II Dataset
1	CVFDT	
2		
3	Online Boosting	
3		

4		
Accuracy Weighted Ensemble (AWE)		
5		
Dynamic Weighted Majority (DWM)		
6		
Hoeffding Option Tree		
7		
PEGASOS (Primal Estimated sub-Gradient Solver for SVM)		
8		
CVFDT _{Nbc}		
9		
Paired learning		
10		
ADWIN Bagging		
11		
ASHT (Adaptive-Size Hoeffding Tree) Bagging		

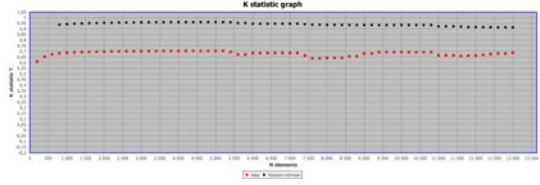
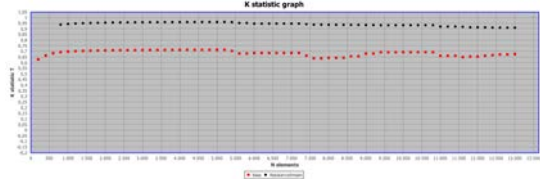
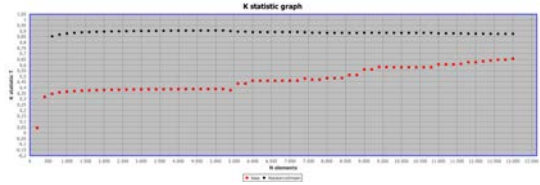
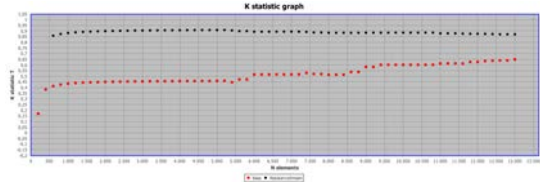
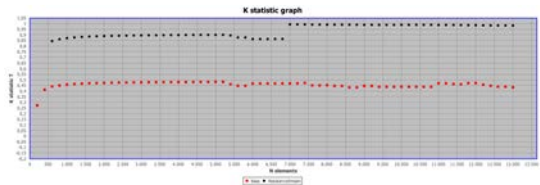
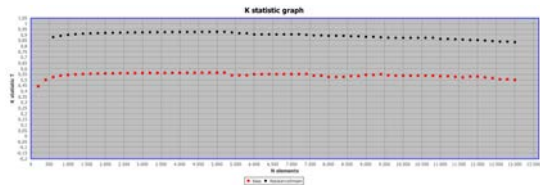
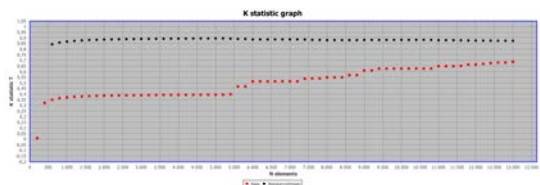
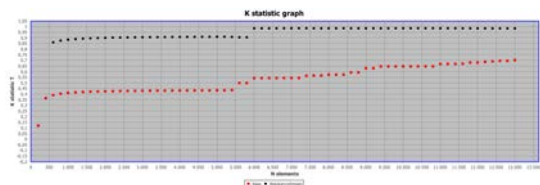
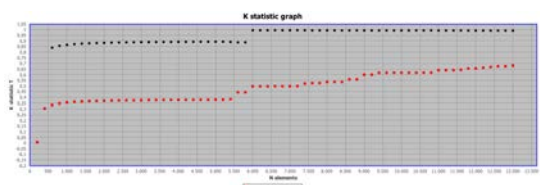
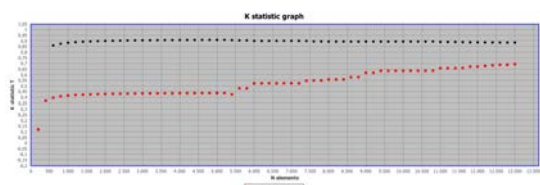
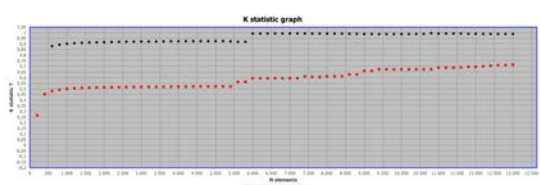
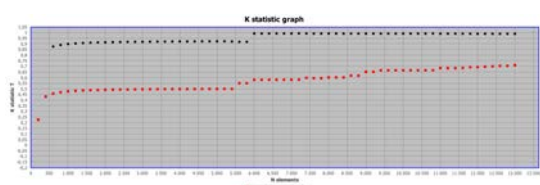
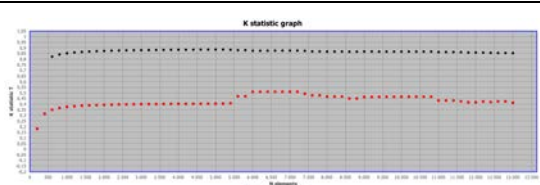
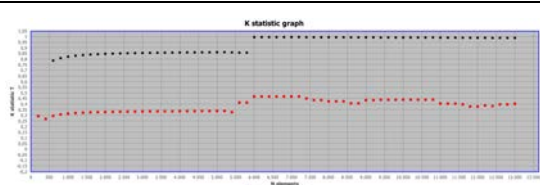
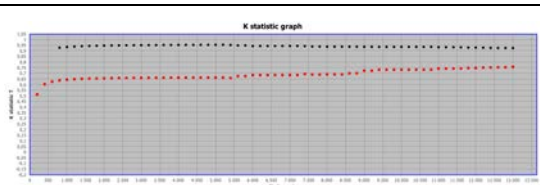
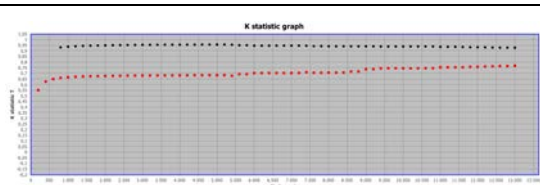
12	Hoeffding Adaptive Tree		
13			
14	Online Coordinate Boosting		
15			
16	Leveraging Bagging		
17			
18	ADAGRAD		
19			
20	Learn++_NSE		
21			
22	OSBoost (Online Smooth Boost)		
23			
24	Accuracy Updated Ensemble (AUE2)		
25			
26	Dynamic Adaptation to Concept Changes (DACC)		
27			

20		
Recurring Concept Drifts (RCD)		
21		
ADOB (Adaptable Diversity-based Online Boosting)		
22		
Boosting-like Online Learning Ensemble (BOLE)		
23		
Hoeffding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)		
24		
SAM-kNN (Self Adjusting Memory model for the k Nearest Neighbor)		
25		
Adaptive Random Forest (ARF)		

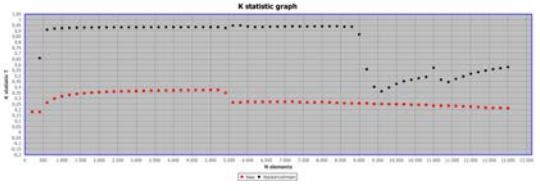
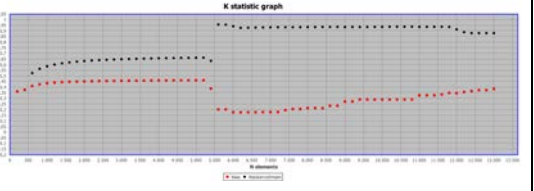
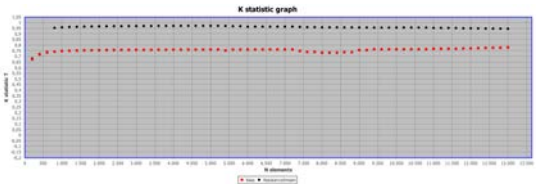
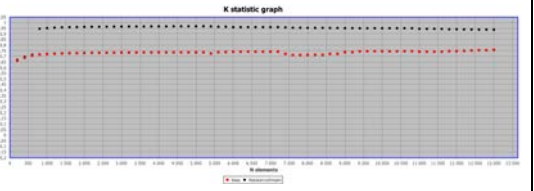
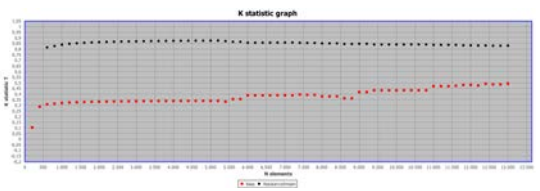
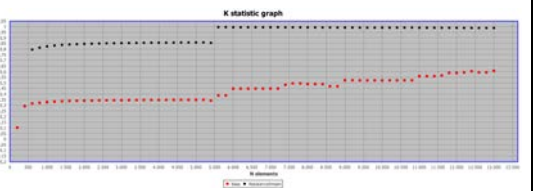
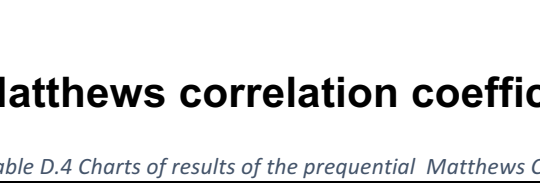
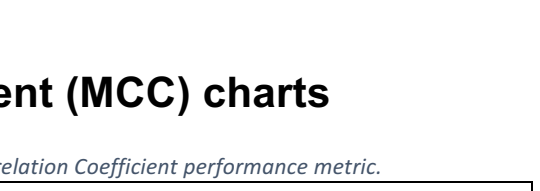
D.3 Cohen's Kappa coefficient charts

Table D.3 Charts of results of the prequential Kappa statistic performance metric.

Algorithm	SwitchCore-I Dataset	SwitchCore-II Dataset
1		
CVFDT		
2		
Online Bagging		
3		
Online Boosting		
4		
Accuracy Weighted Ensemble (AWE)		
5		
Dynamic Weighted Majority (DWM)		
6		
Hoeffding Option Tree		

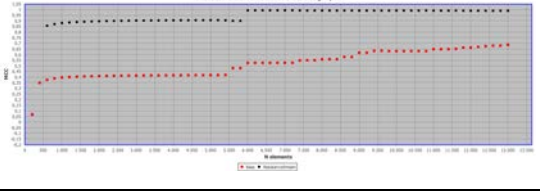
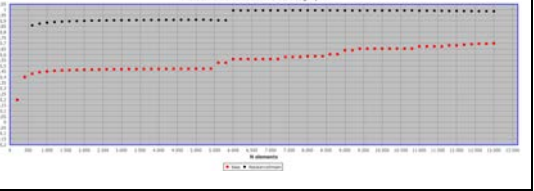
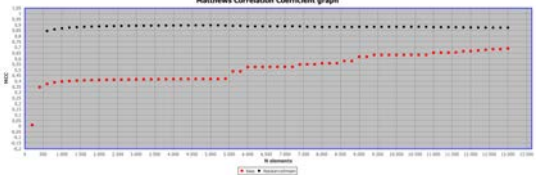



7	PEGASOS (Primal Estimated sub-Gradient solver for SVM)		
8	CVFDT _{lic}		
9	Paired learning		
10	ADWIN Bagging		
11	ASHT (Adaptive-Size Hoeffding Tree) Bagging		
12	Hoeffding Adaptive Tree		
13	Online Coordinate Boosting		
14	Leveraging Bagging		

15	ADAGRAD		
16	Learn++-NSE		
17	OSBoost (Online Smooth Boost)		
18	Accuracy Updated Ensemble (AUE2)		
19	Dynamic Adaptation to Concept Changes (DACC)		
20	Recurring Concept Drifts (RCD)		
21	ADOB (Adaptable Diversity-based Online Boosting)		
22	Boosting-like Online Learning Ensemble (BOLE)		

23	Hoefding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)		
			
24	SAM-kNN (Self Adjusting Memory model for the k Nearest Neighbor)		
25	Adaptive Random Forest (ARF)		

D.4 Matthews correlation coefficient (MCC) charts

Table D.4 Charts of results of the prequential Matthews Correlation Coefficient performance metric.

Algorithm	SwitchCore-I Dataset	SwitchCore-II Dataset
1		
2		
3		

4	Accuracy Weighted Ensemble (AWE)		
5	Dynamic Weighted Majority (DWM)		
6	Hoeffding Option Tree		
7	PEGASOS (Primal Estimated sub-Gradient Solver for SVM)		
8	CVFDT _{NBC}		
9	Paired learning		
10	ADWIN Bagging		
11	ASHT (Adaptive-Size Hoeffding Tree) Bagging		

12		
	Hoefding Adaptive Tree	
13		
	Online Coordinate Boosting	
14		
	Leveraging Bagging	
15		
	ADAGRAD	
16		
	Learn++-NSE	
17		
	OSBoost (Online Smooth Boost)	
18		
	Accuracy Updated Ensemble (AUE2)	
19		
	Dynamic Adaptation to Concept Changes (DACC)	

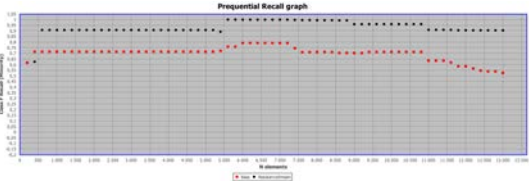

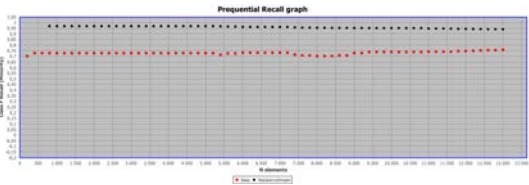
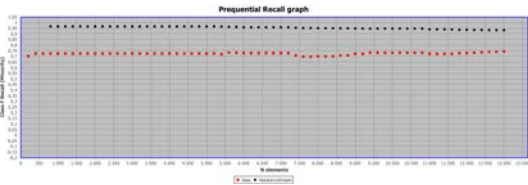
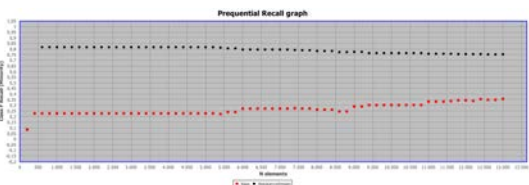
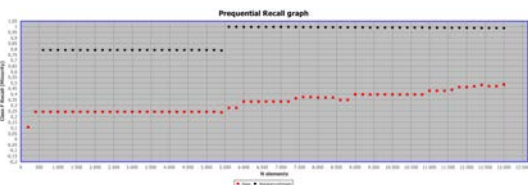
20			
21	Recurring Concept Drifts (RCD)		
22	ADOB (Adaptable Diversity-based Online Boosting)		
23	Boosting-like Online Learning Ensemble (BOLE)		
24	Hoeffding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)		
25	SAM-kNN (Self Adjusting Memory model for the k Nearest Neighbor)		
	Adaptive Random Forest (ARF)		

D.5 Recall or sensitivity charts

Table D.5 Charts of results of the prequential Sensitivity/Recall performance metric.

Algorithm	SwitchCore-I Dataset	SwitchCore-II Dataset
1		
CVFDT		
2		
Online Bagging		
3		
Online Boosting		
4		
Accuracy Weighted Ensemble (AWE)		
5		
Dynamic Weighted Majority (DWM)		
6		
Hoeffding Option Tree		

7	PEGASOS (Primal Estimated sub-Gradient Solver for SVM)		
8	CVFDT _{HC}		
9	Paired learning		
10	ADWIN Bagging		
11	ASHT (Adaptive-Size Hoeffding Tree) Bagging		
12	Hoeffding Adaptive Tree		
13	Online Coordinate Boosting		
14	Leveraging Bagging		

23	Hoeffding Adaptive Tree (HAT) with feature weighted kNN (HAT-kNN-FW) and NB (HAT-kNN-NB)		
24			
25	Adaptive Random Forest (ARF)		

REFERENCES

- Adrichem, N. L. M. van, Asten, B. J. van, & Kuipers, F. A. (2014). Fast Recovery in Software-Defined Networks. *2014 Third European Workshop on Software Defined Networks*, 61–66. <https://doi.org/10.1109/EWSDN.2014.13>
- Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262, 134–147. <https://doi.org/10.1016/j.neucom.2017.04.070>
- Ahmed, A. A., Jantan, A., & Wan, T.-C. (2016). Filtration model for the detection of malicious traffic in large-scale networks. *Computer Communications*, 82, 59–70. <https://doi.org/10.1016/j.comcom.2015.10.012>
- Amershi, S., Lee, B., Kapoor, A., Mahajan, R., & Christian, B. (2011). CueT: Human-guided fast and accurate network alarm triage. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 157–166. <https://doi.org/10.1145/1978942.1978966>
- Arendt, D. L., Burtner, R., Best, D. M., Bos, N. D., Gersh, J. R., Piatko, C. D., & Paul, C. L. (2015). Ocelot: User-centered design of a decision support visualization for network quarantine. *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 1–8. <https://doi.org/10.1109/VIZSEC.2015.7312763>
- Asghar, M. Z., Hämäläinen, S., & Ristaniemi, T. (2012). Self-healing framework for LTE networks. *2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 159–161. <https://doi.org/10.1109/CAMAD.2012.6335320>
- Bach, S. H., & Maloof, M. A. (2008). Paired Learners for Concept Drift. *2008 Eighth IEEE International Conference on Data Mining*, 23–32. <https://doi.org/10.1109/ICDM.2008.119>
- Barddal, J. P., Murilo Gomes, H., Enembreck, F., Pfahringer, B., & Bifet, A. (2016). On Dynamic Feature Weighting for Feature Drifting Data Streams. In P. Frasconi, N. Landwehr, G. Manco, & J. Vreeken (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 129–144). Springer International Publishing. https://doi.org/10.1007/978-3-319-46227-1_9
- Barros, R. S. M. d, Santos, S. G. T. de C., & Júnior, P. M. G. (2016). A Boosting-like Online Learning Ensemble. *2016 International Joint Conference on Neural Networks (IJCNN)*, 1871–1878. <https://doi.org/10.1109/IJCNN.2016.7727427>
- Barros, R., & Santos, S. (2018). A Large-scale Comparison of Concept Drift Detectors. *Information Sciences*, 451–452. <https://doi.org/10.1016/j.ins.2018.04.014>
- Bayer, R. (1972). Symmetric binary B-Trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4), 290–306. <https://doi.org/10.1007/BF00289509>
- Bernardo, A. (2019, June 10). *Incremental rebalancing learning on evolving data streams* [Internet]. <https://es.slideshare.net/AlessioBernardo/incremental-rebalancing-learning-on-evolving-data-streams-149138984>
- Bernardo, A., Valle, E. D., & Bifet, A. (2019). Rebalancing Learning on Evolving Data Streams. *ArXiv:1911.07361*.
- Bifet, A., & Frank, E. (2010). Sentiment Knowledge Discovery in Twitter Streaming Data. In B. Pfahringer, G. Holmes, & A. Hoffmann (Eds.), *Discovery Science* (pp. 1–15). Springer. https://doi.org/10.1007/978-3-642-16184-1_1
- Bifet, A., Frank, E., Holmes, G., & Pfahringer, B. (2010). Accurate Ensembles for Data Streams: Combining Restricted Hoeffding Trees using Stacking. In M. Sugiyama & Q. Yang (Eds.), *Proceedings of 2nd Asian Conference on Machine Learning* (Vol. 13, pp. 225–240). JMLR

- Workshop and Conference Proceedings. <http://proceedings.mlr.press/v13/bifet10a.html>
- Bifet, A., Gavalda, R., Holmes, G., & Pfahringer, B. (2018). *Machine Learning for Data Streams: With Practical Examples in MOA*. The MIT Press.
- Bifet, A., & Gavalda, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (Vol. 1–0, pp. 443–448). Society for Industrial and Applied Mathematics. <https://doi.org/10.1137/1.9781611972771.42>
- Bifet, A., & Gavalda, R. (2009). Adaptive Learning from Evolving Data Streams. In N. M. Adams, C. Robardet, A. Siebes, & J.-F. Boulicaut (Eds.), *Advances in Intelligent Data Analysis VIII* (pp. 249–260). Springer. https://doi.org/10.1007/978-3-642-03915-7_22
- Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11, 1601–1604.
- Bifet, A., Holmes, G., & Pfahringer, B. (2010). Leveraging Bagging for Evolving Data Streams. In J. L. Balcázar, F. Bonchi, A. Gionis, & M. Sebag (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 135–150). Springer. https://doi.org/10.1007/978-3-642-15880-3_15
- Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavalda, R. (2009). New ensemble methods for evolving data streams. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 139–148. <https://doi.org/10.1145/1557019.1557041>
- Bifet, A., Morales, G., Read, J., Holmes, G., & Pfahringer, B. (2015). *Efficient Online Evaluation of Big Data Stream Classifiers* (p. 68). <https://doi.org/10.1145/2783258.2783372>
- Bombal, D., & Duponchelle, J. (2019, January 25). *Getting Started with GNS3 | GNS3 Documentation*. GNS3 Documentation. <https://docs.gns3.com/docs/>
- Boughorbel, S., Jarray, F., & El-Anbari, M. (2017). Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLoS ONE*, 12(6). <https://doi.org/10.1371/journal.pone.0177678>
- Bramer, M. (2013). *Principles of Data Mining* (2nd ed.). Springer Publishing Company, Incorporated.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140. <https://doi.org/10.1007/BF00058655>
- Brezula, R. (2017, September 7). Enterprise Network on GNS3—Part 1—Introduction. *Brezular's Blog*. <http://brezular.com/2017/09/07/enterprise-network-on-gns3-part-1-introduction/>
- Brzezinski, D., & Stefanowski, J. (2014). Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1), 81–94. <https://doi.org/10.1109/TNNLS.2013.2251352>
- Brzezinski, D., & Stefanowski, J. (2017). Prequential AUC: Properties of the area under the ROC curve for data streams with concept drift. *Knowledge and Information Systems*, 52(2), 531–562. <https://doi.org/10.1007/s10115-017-1022-8>
- Bu, K., Wen, X., Yang, B., Chen, Y., Li, E. L., & Chen, X. (2016). Is every flow on the right track?: Inspect SDN forwarding with RuleScope. *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. <https://doi.org/10.1109/INFOCOM.2016.7524333>
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0 Step-by-step data mining guide*. The CRISP-DM consortium. <https://maestria-datamining-2010.googlecode.com/svn-history/r282/trunk/dmct-teorica/tp1/CRISPWP-0800.pdf>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- Chen, S.-T., Lin, H.-T., & Lu, C.-J. (2012). An online boosting algorithm with theoretical justifications. *Proceedings of the 29th International Conference on Machine Learning*

- Learning*, 1873–1880.
- Cheng, L., Qiu, X., Meng, L., Qiao, Y., & Boutaba, R. (2010). Efficient Active Probing for Fault Diagnosis in Large Scale and Noisy Networks. *2010 Proceedings IEEE INFOCOM*, 1–9. <https://doi.org/10.1109/INFCOM.2010.5462041>
- Cheng, S., Cheng, Z., Luan, Z., & Qian, D. (2011). NEPnet: A scalable monitoring system for anomaly detection of network service. *2011 7th International Conference on Network and Service Management*, 1–5.
- Chhabra, P., Scott, C., Kolaczyk, E. D., & Crovella, M. (2008). Distributed Spatial Anomaly Detection. *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 1705–1713. <https://doi.org/10.1109/INFCOM.2008.232>
- Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1), 6. <https://doi.org/10.1186/s12864-019-6413-7>
- Cho, H. C., Fadali, S. M., & Lee, H. (2008). Adaptive neural queue management for TCP networks. *Computers & Electrical Engineering*, 34(6), 447–469. <https://doi.org/10.1016/j.compeleceng.2008.02.002>
- Cisco Academy. (2014). *Connecting Networks Companion Guide, Video Enhanced Edition* (1st ed.). Cisco Press.
- Claise, B. (2004). *Cisco Systems NetFlow Services Export Version 9* (RFC No. 3954). RFC Editor. <http://www.rfc-editor.org/rfc/rfc3954.txt>
- Clark-Carter, D. (2010). Measures of Central Tendency. In P. Peterson, E. Baker, & B. McGaw (Eds.), *International Encyclopedia of Education (Third Edition)* (pp. 264–266). Elsevier. <https://doi.org/10.1016/B978-0-08-044894-7.01343-9>
- Cobo, M. J., López-Herrera, A. G., Herrera-Viedma, E., & Herrera, F. (2011). An approach for detecting, quantifying, and visualizing the evolution of a research field: A practical application to the Fuzzy Sets Theory field. *Journal of Informetrics*, 5(1), 146–166. <https://doi.org/10.1016/j.joi.2010.10.002>
- Cobo, M. J., López-Herrera, A. G., Herrera-Viedma, E., & Herrera, F. (2012). SciMAT: A new science mapping analysis software tool. *Journal of the American Society for Information Science and Technology*, 63(8), 1609–1630. <https://doi.org/10.1002/asi.22688>
- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1), 37–46. <https://doi.org/10.1177/001316446002000104>
- Colman-Meixner, C., Develder, C., Tornatore, M., & Mukherjee, B. (2016). A Survey on Resiliency Techniques in Cloud Computing Infrastructures and Applications. *IEEE Communications Surveys Tutorials*, 18(3), 2244–2281. <https://doi.org/10.1109/COMST.2016.2531104>
- Conlan, P. J. (2009). *Cisco Network Professional's Advanced Internetworking Guide*. SYBEX Inc.
- Coulter, N., Monarch, I., & Konda, S. (1998). Software engineering as seen through its research literature: A study in co-word analysis. *Journal of the American Society for Information Science*, 49(13), 1206–1223. [https://doi.org/10.1002/\(SICI\)1097-4571\(1998\)49:13<1206::AID-ASI7>3.0.CO;2-F](https://doi.org/10.1002/(SICI)1097-4571(1998)49:13<1206::AID-ASI7>3.0.CO;2-F)
- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27. <https://doi.org/10.1109/TIT.1967.1053964>
- Del Campo-Ávila, J. (2007). *Nuevos enfoques en aprendizaje incremental* [Universidad de Málaga]. <https://riuma.uma.es/xmlui/handle/10630/6902>
- Delgado, R., & Tibau, X.-A. (2019). Why Cohen's Kappa should be avoided as performance measure in classification. *PLOS ONE*, 14(9), e0222916. <https://doi.org/10.1371/journal.pone.0222916>
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12(null), 2121–2159.

- Duenas, J. C., Navarro, J. M., Parada G., H. A., Andion, J., & Cuadrado, F. (2018). Applying Event Stream Processing to Network Online Failure Prediction. *IEEE Communications Magazine*, 56(1), 166–170. <https://doi.org/10.1109/MCOM.2018.1601135>
- Dusia, A., & Sethi, A. S. (2016a). Recent Advances in Fault Localization in Computer Networks. *IEEE Communications Surveys Tutorials*, 18(4), 3030–3051. <https://doi.org/10.1109/COMST.2016.2570599>
- Dusia, A., & Sethi, A. S. (2016b). Recent Advances in Fault Localization in Computer Networks. *IEEE Communications Surveys Tutorials*, 18(4), 3030–3051. <https://doi.org/10.1109/COMST.2016.2570599>
- Elwell, R., & Polikar, R. (2011). Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks*, 22(10), 1517–1531. <https://doi.org/10.1109/TNN.2011.2160459>
- Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018a). Learning from Imbalanced Data Streams. In *Learning from Imbalanced Data Sets* (pp. 279–303). Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4_11
- Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018b). *Learning from Imbalanced Data Sets*. Springer.
- Flach, P. A. (2010). ROC Analysis. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 869–875). Springer US. https://doi.org/10.1007/978-0-387-30164-8_733
- Fraïwan, M., & Manimaran, G. (2008). Localization of IP Links Faults Using Overlay Measurements. *2008 IEEE International Conference on Communications*, 5629–5633. <https://doi.org/10.1109/ICC.2008.1055>
- Frank, E., & Witten, I. H. (1998). Generating Accurate Rule Sets Without Global Optimization. *Proceedings of the Fifteenth International Conference on Machine Learning*, 144–151.
- Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, 148–156.
- García-Algarra, J., Arozarena, P., García-Gómez, S., Carrera-Barroso, A., & Toribio-Sardón, R. (2011). A Lightweight Approach to Distributed Network Diagnosis under Uncertainty. In S. Caballé, F. Xhafa, & A. Abraham (Eds.), *Intelligent Networking, Collaborative Systems and Applications* (pp. 95–116). Springer. https://doi.org/10.1007/978-3-642-16793-5_5
- Gerhards, R. (2009). *The Syslog Protocol* (RFC No. 5424). RFC Editor. <http://www.rfc-editor.org/rfc/rfc5424.txt>
- Gill, P., Jain, N., & Nagappan, N. (2011). Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. *Proceedings of the ACM SIGCOMM 2011 Conference*, 350–361. <https://doi.org/10.1145/2018436.2018477>
- Gillani, S. F., Demirci, M., Al-Shaer, E., & Ammar, M. H. (2014). Problem Localization and Quantification Using Formal Evidential Reasoning for Virtual Networks. *IEEE Transactions on Network and Service Management*, 11(3), 307–320. <https://doi.org/10.1109/TNSM.2014.2326297>
- Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A Survey on Ensemble Learning for Data Stream Classification. *ACM Computing Surveys*, 50(2), 23:1-23:36. <https://doi.org/10.1145/3054925>
- Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9), 1469–1495. <https://doi.org/10.1007/s10994-017-5642-8>
- Gonçalves Jr, P. M., & Barros, R. S. M. de. (2013). RCD: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9), 1018–1025. <https://doi.org/10.1016/j.patrec.2013.02.005>

- Guibas, L. J., & Sedgewick, R. (1978). A dichromatic framework for balanced trees. *19th Annual Symposium on Foundations of Computer Science (Sfcs 1978)*, 8–21. <https://doi.org/10.1109/SFCS.1978.3>
- Guo Jiangwei, Wu Xiaoping, & Ye Qing. (2010). Network fault diagnosis based on rough set-support vector machine. *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 14, V14-312-V14-315. <https://doi.org/10.1109/ICCASM.2010.5622291>
- Hanemann, A. (2006). A hybrid rule-based/case-based reasoning approach for service fault diagnosis. *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, 2, 5 pp.-. <https://doi.org/10.1109/AINA.2006.29>
- Harvey, N. J. A., Patrascu, M., Wen, Y., Yekhanin, S., & Chan, V. W. S. (2007). Non-Adaptive Fault Diagnosis for All-Optical Networks via Combinatorial Group Testing on Graphs. *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 697–705. <https://doi.org/10.1109/INFCOM.2007.87>
- He, W., Hu, G., & Zhou, Y. (2012). Large-scale IP network behavior anomaly detection and identification using substructure-based approach and multivariate time series mining. *Telecommunication Systems*, 50(1), 1–13. <https://doi.org/10.1007/s11235-010-9384-1>
- Huang, X., Zou, S., Wang, W., & Cheng, S. (2006). Fault management for Internet Services: Modeling and Algorithms. *2006 IEEE International Conference on Communications*, 2, 854–859. <https://doi.org/10.1109/ICC.2006.254814>
- Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 97–106. <https://doi.org/10.1145/502512.502529>
- Huynh, M., Mohapatra, P., & Goose, S. (2007). Cross-over spanning trees Enhancing metro ethernet resilience and load balancing. *2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS '07)*, 251–260. <https://doi.org/10.1109/BROADNETS.2007.4550433>
- Jaber, G., Cornuéjols, A., & Tarroux, P. (2013). A New On-Line Learning Method for Coping with Recurring Concepts: The ADACC System. In M. Lee, A. Hirose, Z.-G. Hou, & R. M. Kil (Eds.), *Neural Information Processing* (pp. 595–604). Springer. https://doi.org/10.1007/978-3-642-42042-9_74
- Jian, L. (2010). Heuristic Fault Localization Algorithm Based on Bayesian Suspected Degree. *Journal of Software*.
- John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 338–345.
- Johnsson, A., & Meirosu, C. (2013). Towards automatic network fault localization in real time using probabilistic inference. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 1393–1398.
- Johnsson, A., Meirosu, C., & Flinta, C. (2014). Online network performance degradation localization using probabilistic inference and change detection. *2014 IEEE Network Operations and Management Symposium (NOMS)*, 1–8. <https://doi.org/10.1109/NOMS.2014.6838255>
- Kamisiński, A., & Fung, C. (2015). FlowMon: Detecting Malicious Switches in Software-Defined Networks. *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, 39–45. <https://doi.org/10.1145/2809826.2809833>
- Kanda, Y., Fontugne, R., Fukuda, K., & Sugawara, T. (2013). ADMIRE: Anomaly detection method using entropy-based PCA with three-step sketches. *Computer Communications*, 36(5), 575–588. <https://doi.org/10.1016/j.comcom.2012.12.002>
- Kasai, H., Kellerer, W., & Kleinsteuher, M. (2016). Network Volume Anomaly Detection and

- Identification in Large-Scale Networks Based on Online Time-Structured Traffic Tensor Tracking. *IEEE Transactions on Network and Service Management*, 13(3), 636–650. <https://doi.org/10.1109/TNSM.2016.2598788>
- Kavulya, S. P., Daniels, S., Joshi, K., Hiltunen, M., Gandhi, R., & Narasimhan, P. (2012). Draco: Statistical Diagnosis of Chronic Problems in Large Distributed Systems. *Proceedings of the 2012 42Nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 1–12. <http://dl.acm.org/citation.cfm?id=2354410.2355155>
- Khan, M. A., Peters, S., Sahinel, D., Pozo-Pardo, F. D., & Dang, X.-T. (2018). Understanding autonomic network management: A look into the past, a solution for the future. *Computer Communications*, 122, 93–117. <https://doi.org/10.1016/j.comcom.2018.01.014>
- Kirman, E., & Hood, C. S. (2004). Diagnosing network states through intelligent probing. *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)*, 1, 147–160 Vol.1. <https://doi.org/10.1109/NOMS.2004.1317651>
- Kolter, J. Z., & Maloof, M. A. (2007). Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. *The Journal of Machine Learning Research*, 8, 2755–2790.
- Krishnamurthy, B., Sen, S., Zhang, Y., & Chen, Y. (2003). Sketch-based change detection: Methods, evaluation, and applications. *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, 234–247. <https://doi.org/10.1145/948205.948236>
- Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2), 195–215. <https://doi.org/10.1023/A:1007452223027>
- Kubat, M., Holte, R., & Matwin, S. (1997). Learning when negative examples abound. In M. van Someren & G. Widmer (Eds.), *Machine Learning: ECML-97* (pp. 146–153). Springer. https://doi.org/10.1007/3-540-62858-4_79
- Li, C., Liu, L., & Pang, X. (2009). A Dynamic Probability Fault Localization Algorithm Using Digraph. *2009 Fifth International Conference on Natural Computation*, 6, 187–191. <https://doi.org/10.1109/ICNC.2009.501>
- Li, T., & Li, X. (2010). Novel alarm correlation analysis system based on association rules mining in telecommunication networks. *Information Sciences: An International Journal*, 180(16), 2960–2978. <https://doi.org/10.1016/j.ins.2010.04.013>
- Li, Z., Cheng, L., Qiu, X., & Zeng, Y. (2009). Fault Diagnosis for Large-Scale IP Networks Based on Dynamic Bayesian Model. *2009 Fifth International Conference on Natural Computation*, 6, 67–71. <https://doi.org/10.1109/ICNC.2009.246>
- Li, Z., Huang, W., Xiong, Y., Ren, S., & Zhu, T. (2020). Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. *Knowledge-Based Systems*, 195, 105694. <https://doi.org/10.1016/j.knosys.2020.105694>
- Likun Yu, Lu Cheng, Yan Qiao, Yiguo Yuan, & Xingyu Chen. (2010). An efficient active probing approach based on the combination of online and offline strategies. *2010 International Conference on Network and Service Management*, 298–301. <https://doi.org/10.1109/CNSM.2010.5691213>
- Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., & Chen, X. (2016). Log Clustering Based Problem Identification for Online Service Systems. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 102–111.
- Loshin, D. (2011). Dimensions of Data Quality. In *The Practitioner's Guide to Data Quality Improvement* (pp. 129–146). Elsevier. <https://doi.org/10.1016/B978-0-12-373717-5.00008-7>
- Losing, V., Hammer, B., & Wersing, H. (2016). KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 291–300. <https://doi.org/10.1109/ICDM.2016.0040>

- Lu, L., Xu, Z., Wang, W., & Sun, Y. (2013). A new fault detection method for computer networks. *Reliability Engineering & System Safety*, 114, 45–51. <https://doi.org/10.1016/j.ress.2012.12.015>
- Mahimkar, A. A., Ge, Z., Shaikh, A., Wang, J., Yates, J., Zhang, Y., & Zhao, Q. (2009). Towards Automated Performance Diagnosis in a Large IPTV Network. *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, 231–242. <https://doi.org/10.1145/1592568.1592596>
- Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2), 442–451. [https://doi.org/10.1016/0005-2795\(75\)90109-9](https://doi.org/10.1016/0005-2795(75)90109-9)
- Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M., & Tornatore, M. (2019). An Overview on Application of Machine Learning Techniques in Optical Networks. *IEEE Communications Surveys Tutorials*, 21(2), 1383–1408. <https://doi.org/10.1109/COMST.2018.2880039>
- Nagios Enterprises. LLC. (2021). *Nagios Library*. Nagios - The Industry Standard In IT Infrastructure Monitoring. <https://library.nagios.com/>
- Narayanan, H. T. S., Ilangovan, G., & Narayanan, S. (2013). Feasibility of SNMP OID compression. *Journal of King Saud University - Computer and Information Sciences*, 25(1), 35–42. <https://doi.org/10.1016/j.jksuci.2012.05.006>
- Natu, M., & Sethi, A. S. (2008a). Using temporal correlation for fault localization in dynamically changing networks. *International Journal of Network Management*, 18(4), 303–316. <https://doi.org/10.1002/nem.659>
- Natu, M., & Sethi, A. S. (2006). Active Probing Approach for Fault Localization in Computer Networks. *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, 25–33. <https://doi.org/10.1109/E2EMON.2006.1651276>
- Natu, M., & Sethi, A. S. (2008b). Application of adaptive probing for fault diagnosis in computer networks. *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, 1055–1060. <https://doi.org/10.1109/NOMS.2008.4575278>
- Nováczki, S. (2013). An improved anomaly detection and diagnosis framework for mobile network operators. *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN)*, 234–241.
- Oza, N. C., & Russell, S. (2001). Online Bagging and Boosting. *In Artificial Intelligence and Statistics 2001*, 105–112.
- Ozcelik, B., & Yilmaz, C. (2017). Seer: A Lightweight Online Failure Prediction Approach. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 1, 624–625. <https://doi.org/10.1109/COMPSAC.2017.210>
- Pandora FMS. (2020). *Documentación—Pandora FMS* [Pandora FMS]. <https://pandorafms.com/es/community/pandora-docs/>
- Patil, M. M. (2019). Handling Concept Drift in Data Streams by Using Drift Detection Methods. In V. E. Balas, N. Sharma, & A. Chakrabarti (Eds.), *Data Management, Analytics and Innovation* (pp. 155–166). Springer Singapore.
- Pelossof, R., Jones, M., Vovsha, I., & Rudin, C. (2008). Online Coordinate Boosting. *ArXiv:0810.4553 [Stat]*. <http://arxiv.org/abs/0810.4553>
- Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New Options for Hoeffding Trees. In M. A. Orgun & J. Thornton (Eds.), *AI 2007: Advances in Artificial Intelligence* (pp. 90–99). Springer. https://doi.org/10.1007/978-3-540-76928-6_11
- Potharaju, R., & Jain, N. (2013). When the Network Crumbles: An Empirical Study of Cloud Network Failures and Their Impact on Services. *Proceedings of the 4th Annual Symposium on Cloud*

- Computing, 15:1-15:17. <https://doi.org/10.1145/2523616.2523638>
- Potharaju, R., Jain, N., & Nita-Rotaru, C. (2013). Juggling the Jigsaw: Towards automated problem inference from network trouble tickets. *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, 127–142.
- Prieto, A. G., Gillblad, D., Steinert, R., & Miron, A. (2011). Toward decentralized probabilistic management. *IEEE Communications Magazine*, 49(7), 80–86. <https://doi.org/10.1109/MCOM.2011.5936159>
- Priya, S., & Uthra, R. A. (2020). Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. *Journal of Ambient Intelligence and Humanized Computing*. <https://doi.org/10.1007/s12652-020-01934-y>
- Qi, J., Wu, F., Li, L., & Shu, H. (2007). Artificial intelligence applications in the telecommunications industry. *Expert Systems*, 24(4), 271–291. <https://doi.org/10.1111/j.1468-0394.2007.00433.x>
- Realì, G., & Monacelli, L. (2009). Definition and performance evaluation of a fault localization technique for an NGN IMS network. *IEEE Transactions on Network and Service Management*, 6(2), 122–136. <https://doi.org/10.1109/TNSM.2009.090605>
- Rish, I., Brodie, M., Sheng Ma, Odintsova, N., Beygelzimer, A., Grabarnik, G., & Hernandez, K. (2005). Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks*, 16(5), 1088–1109. <https://doi.org/10.1109/TNN.2005.853423>
- Ritter, H., Winter, R., & Schiller, J. (2004). A partition detection system for mobile ad-hoc networks. *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, 489–497. <https://doi.org/10.1109/SAHCN.2004.1381951>
- Salah, S., Maciá-Fernández, G., & Díaz-Verdejo, J. E. (2013). A model-based survey of alert correlation techniques. *Computer Networks*, 57(5), 1289–1317. <https://doi.org/10.1016/j.comnet.2012.10.022>
- Salzberg, S. L. (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 16(3). <https://doi.org/10.1007/BF00993309>
- Sammut, C., & Webb, G. I. (Eds.). (2010a). Area Under Curve. In *Encyclopedia of Machine Learning* (pp. 40–40). Springer US. https://doi.org/10.1007/978-0-387-30164-8_28
- Sammut, C., & Webb, G. I. (Eds.). (2010b). Sensitivity. In *Encyclopedia of Machine Learning* (pp. 901–901). Springer US. https://doi.org/10.1007/978-0-387-30164-8_751
- Santos, S. G. T. de C., Gonçalves Júnior, P. M., Silva, G. D. dos S., & de Barros, R. S. M. (2014). Speeding Up Recovery from Concept Drifts. In T. Calders, F. Esposito, E. Hüllermeier, & R. Meo (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 179–194). Springer. https://doi.org/10.1007/978-3-662-44845-8_12
- Sauvanaud, C., Kaâniche, M., Kanoun, K., Lazri, K., & Silvestre, G. (2018). Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned. *J. Syst. Softw.* <https://doi.org/10.1016/j.jss.2018.01.039>
- Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOLver for SVM. *Proceedings of the 24th International Conference on Machine Learning*, 807–814. <https://doi.org/10.1145/1273496.1273598>
- Sharma, P., Banerjee, S., Tandel, S., Aguiar, R., Amorim, R., & Pinheiro, D. (2013). Enhancing network management frameworks with SDN-like control. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 688–691.
- Steinder, M., Igorzata, & Sethi, A. S. (2004a). A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 53(2), 165–194. <https://doi.org/10.1016/j.scico.2004.01.010>
- Steinder, M., & Sethi, A. S. (2004b). Probabilistic fault diagnosis in communication systems through

- incremental hypothesis updating. *Computer Networks*, 45(4), 537–562. <https://doi.org/10.1016/j.comnet.2004.01.007>
- Steinder, M., & Sethi, A. S. (2004c). Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5), 809–822. <https://doi.org/10.1109/TNET.2004.836121>
- Steinert, R., Gestrelus, S., & Gillblad, D. (2011). A Distributed Spatio-Temporal Event Correlation Protocol for Multi-Layer Virtual Networks. *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 1–5. <https://doi.org/10.1109/GLOCOM.2011.6133988>
- Steinert, R., & Gillblad, D. (2010). Long-Term Adaptation and Distributed Detection of Local Network Changes. *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 1–5. <https://doi.org/10.1109/GLOCOM.2010.5684137>
- Szilagyi, P., & Novaczki, S. (2012). An Automatic Detection and Diagnosis Framework for Mobile Communication Systems. *IEEE Transactions on Network and Service Management*, 9(2), 184–197. <https://doi.org/10.1109/TNSM.2012.031912.110155>
- Takeshita, K., Yokota, M., & Nishimatsu, K. (2015). Early network failure detection system by analyzing Twitter data. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 279–286. <https://doi.org/10.1109/INM.2015.7140302>
- Tang, Y., Cheng, G., Xu, Z., Chen, F., Elmansor, K., & Wu, Y. (2016). Automatic belief network modeling via policy inference for SDN fault localization. *Journal of Internet Services and Applications*, 7(1), 1. <https://doi.org/10.1186/s13174-016-0043-y>
- Teare, D. (2012). *Designing for Cisco Internetwork Solutions (Desgn) Foundation Learning Guide (Paperback): (Authorized Ccda Selfstudy Guide)(Exam 640-863)* (2nd ed.). Pearson Education.
- Ting, K. M. (2010a). Precision. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 780–780). Springer US. https://doi.org/10.1007/978-0-387-30164-8_651
- Ting, K. M. (2010b). Precision and Recall. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 781–781). Springer US. https://doi.org/10.1007/978-0-387-30164-8_652
- Utgoff, P. E. (2017). Incremental Learning. In *Encyclopedia of Machine Learning and Data Mining* (pp. 634–637). https://doi.org/10.1007/978-1-4899-7687-1_130
- Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs. *Proceedings of the 3rd IEEE Workshop on IP Operations Management (IPOM 2003)* (IEEE Cat. No.03EX764), 119–126. <https://doi.org/10.1109/IPOM.2003.1251233>
- Vargas-Arcila, A. M., Corrales, J. C., Sanchis, A., & Rendón, A. (2021). *Dataset of symptom-fault causal relationships for an IP-based network*. <https://doi.org/10.17632/tc6ysmh5j8.1>
- Wang, F., Wang, H., Wang, X., & Su, J. (2012). A new multistage approach to detect subtle DDoS attacks. *Math. Comput. Model.* <https://doi.org/10.1016/j.mcm.2011.02.025>
- Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 226–235. <https://doi.org/10.1145/956750.956778>
- Wang, H., Wang, L., Yu, Q., Zheng, Z., Bouguettaya, A., & Lyu, M. R. (2017). Online Reliability Prediction via Motifs-Based Dynamic Bayesian Networks for Service-Oriented Systems. *IEEE Transactions on Software Engineering*, 43(6), 556–579. <https://doi.org/10.1109/TSE.2016.2615615>
- Wang, H., Wang, Y., Qiu, X., Li, W., & Xiao, A. (2015). Fault diagnosis based on evidences screening in virtual network. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 802–805. <https://doi.org/10.1109/INM.2015.7140380>
- Wang, S., Minku, L. L., & Yao, X. (2013). A learning framework for online class imbalance learning. *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, 36–45. <https://doi.org/10.1109/CIEL.2013.6613138>

- Wang, T., Srivatsa, M., Agrawal, D., & Liu, L. (2010). Spatio-temporal patterns in network events. *Proceedings of the 6th International Conference*, 1–12. <https://doi.org/10.1145/1921168.1921172>
- Watson, M. R., Shirazi, N., Marnerides, A. K., Mauthe, A., & Hutchison, D. (2016). Malware Detection in Cloud Computing Infrastructures. *IEEE Transactions on Dependable and Secure Computing*, 13(2), 192–205. <https://doi.org/10.1109/TDSC.2015.2457918>
- Webb, G. I. (2010). Naïve Bayes. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 713–714). Springer US. https://doi.org/10.1007/978-0-387-30164-8_576
- Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (Eds.). (2017). *Data Mining—Practical Machine Learning Tools and Techniques* (Fourth Edition). Morgan Kaufmann. <https://doi.org/10.1016/B978-0-12-804291-5.00015-5>
- Wu, S., Flach, P., & Ferri, C. (2007). An Improved Model Selection Heuristic for AUC. In J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Machine Learning: ECML 2007* (pp. 478–489). Springer. https://doi.org/10.1007/978-3-540-74958-5_44
- Wu, X., Turner, D., Chen, C.-C., Maltz, D. A., Yang, X., Yuan, L., & Zhang, M. (2012). NetPilot: Automating datacenter network failure mitigation. *ACM SIGCOMM Computer Communication Review*, 42(4), 419–430. <https://doi.org/10.1145/2377677.2377759>
- Xueshan Shan, & Li, J. J. (2001). A case study of IP network monitoring using wireless mobile devices. *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, 590–593. <https://doi.org/10.1109/ICCCN.2001.956328>
- Yan, C., Wang, Y., Qiu, X., Li, W., & Guan, L. (2014). Multi-layer fault diagnosis method in the Network Virtualization Environment. *The 16th Asia-Pacific Network Operations and Management Symposium*, 1–6. <https://doi.org/10.1109/APNOMS.2014.6996580>
- Yang, K., Liu, R., Sun, Y., Yang, J., & Chen, X. (2017). Deep Network Analyzer (DNA): A Big Data Analytics Platform for Cellular Networks. *IEEE Internet of Things Journal*, 4(6), 2019–2027. <https://doi.org/10.1109/JIOT.2016.2624761>
- Yuan, C., Ma, J., Xu, H., Liu, L., & Zhang, J.-G. (2015). Chaos optical time domain reflectometer using a laser modulated by Boolean chaos. *Journal of Optoelectronics-laser*. <https://doi.org/10.16136/j.joel.2015.11.0570>
- Zabbix LLC. (2021). *ZABBIX Documentation*. ZABBIX. <https://www.zabbix.com/manuals>
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). *Dive into Deep Learning*. <https://d2l.ai/>
- Zhou, D., Yan, Z., Fu, Y., & Yao, Z. (2018). A survey on network data collection. *Journal of Network and Computer Applications*, 116, 9–23. <https://doi.org/10.1016/j.jnca.2018.05.004>