ONTIC: D5.4: Use Case #1 Network Intrusion Detection

Miguel-Angel Lopez, José-Maria Ocon, Carlos Area Area Rúa, Juliette Dromard, Philippe Owezarski

Online Network Traffic Characterization

Deliverable
Use Case #1 Network Intrusion Detection

ONTIC Project
(GA number 619633)

Deliverable D5.4
Dissemination Level: PUBLIC

Authors

Miguel Ángel López Peña, José María Ocón Quintana; Carlos Area Rúa
SATEC

Juliette Dromard, Philippe Owezarski
LAAS-CNRS

Version

ONTIC_D5.4_2017.01.31_1.0.DOCX

Version History

Version	Modification date	Modified by	Summary
0.0	2016-09-26	SATEC	Structure proposal
0.1	2016-10-18	SATEC	First draft version
0.2	2016-11-25	CNRS SATEC	Contributions by CNRS and SATEC
0.3	2016-01-13	ADAPTIT CNRS UPM ERC SATEC	Second Draft: First review
0.4	2017-01-30	CNRS SATEC	Corrections after first review
1.0	2017-01-31	SATEC	QA Review Final version

Quality Assurance:

Role	Name
Quality Assurance Manager	**Miguel Ángel López Peña (SATEC)**
Reviewer #1	**Panos Georgatsos (ADAPTIT)**
Reviewer #2	**Alberto Mozo (UPM)**

Table of Contents

List of figures

List of tables

1. Acronyms and Definitions

1.1 Acronyms

Acronym	Defined as
CSP	Communication Service Provider
CSV	Comma Separated Values
DoD	Definition of Done
DoS	Denial of Service
DoW	Description of Work
DTD	Document Type Definition
FNR	False Negative Rate
FPR	False Positive Rate
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
IDGCA	Incremental Density Grid-based Clustering Algorithm
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
ML	Machine Learning
NA	Network Administrator
NM	Network Monitoring
ONTIC	Online Network Traffic Characterization
ONTS	ONTIC Network Traffic Summary
ORUNADA	Online and Real-time Unsupervised Network Anomaly Detection Algorithm
PCAP	Packet Capture
QoE	Quality of Experience
QoS	Quality of Service
RDBMS	Relational Database Management System
REST	Representational State Transfer
SQL	Structured Query Language
TPR	True Positive Rate
UC	Use Case
UML	Unified Modelling Language
US	User Story
XML	Extensible Mark-up Language

2. Purpose of the Document

Deliverable D5.4 purpose is to document the final set of requirements for Use Case #1 (Network Intrusion Detection) jointly with the design, implementation and validation of its corresponding prototype. Additionally, updates in the Use Case requirements are also shown.

The ONTIC Use Case development and implementation follows a customized version of the Scrum Agile methodology (as described in deliverable D5.1 [3]); therefore, the requirements are described as user stories.

The different sections in the document provide:

- Introduction of Use Case #1 in terms of their application in CSP environments, operational goals and machine learning algorithms (section 7)
- Use case #1 specification is described in section 8). Definitions of Done (DoD) are provided in (Annex A)
- Use case #1 design is detailed in section 9.
- Use case #1 implementation details are documented in section 10 .
- Use case #1 testing documentation is shown in section 11

3. Scope

This document provides information about Use Case requirements (as user stories) and corresponding prototype implementation. Therefore, it is not expected to provide description of algorithms descriptions or a description of the ONTIC Big Data Architecture, as there are specific deliverables for said topics (D3.2 and D4.2, and D2.3, respectively), unless absolutely needed for the understanding of the Use Case implementation.

4. Intended Audience

The intended document audience includes not only all the partners in the ONTIC consortium (especially those involved in gathering requirements, and in designing, implementing and validating the prototypes) or the receivers of the project. It also includes any reader interested in understanding the ONTIC Use Cases and the business principles that guide the research within the project.

5. Suggested Previous Readings

It is expected that a basic background on Information and Communications Technology (ICT) is sufficient to address the contents of this document; however, some previous readings are suggested:

- ONTIC. “Deliverable D4.2. Algorithms Description” [1].
- ONTIC. “Deliverable D4.3. Experimental Evaluation of Algorithms for Online Network Characterization” [2].
- ONTIC. “Deliverable D5.1. Use Case Requirements” [3].
- ONTIC. “Deliverable D5.2. Progress on Use Cases [4].

6. Executive Summary

In the context of network management and engineering, ONTIC initially identified (in the DoW) three scenarios to address network issues of increased importance. During the project's first year, the initial Use Cases were refined and assigned a specific slogan: (UC #1) Network Anomaly Detection; (UC #2) Adaptive Quality of Experience (QoE) Control; and (UC #3) Proactive Congestion Detection and Control Systems. During the project's second and third year, the Use Case requirements have been further refined and the functionalities required have been implemented at a system level.

Use Case #1 aims at providing a system able to perform online network traffic monitoring for detecting in real-time network anomalies. For enhancing its applicability in realistic ISP/CSP network environments, Use Case #1 defines a scenario in which automatic anomaly detection is embedded within an administrator-oriented network supervision tool. This way, network traffic and potential anomalies can be analysed and traced back if required through a set of integrated visual interfaces fitting the every-day practices of network administrators. As such, UC #1 encompasses two subsystems: The Anomaly Detection Subsystem and the Analytic Dashboard Subsystem. This Deliverable presents the specification, design, implementation and evaluation of these subsystems.

The results show valid and scalable performance. And equally important, that system design and implementation can sufficiently resolve the critical challenge of timely synchronization –within real-time response constraints- between independent applications and processes that have to work on Big Data and share produced results and alarms.

7. Use Case #1 Environment

7.1 Overall

ONTIC has specified a number of uses cases for exhibiting the application of its results in CSP (Communication Service Providers) environments. The purpose is to show via close-to-the-market Use Cases that the proposed off/on-line machine learning (ML) algorithms for traffic analysis provide effective and efficient solutions to critical problems of concern to CSP's such as network security, congestion avoidance and QoE management.

This chapter outlines the Use Case #1 considered by the project in terms of its application context in CSP environments, operational goals and ML algorithms used. These aspects are summarized in Table 1. A detailed description of the Use Case including system model, evaluation scenarios and current status of development, is presented in chapter 8. A comprehensive review of the relevant state of art was included in the previous versions of the deliverable, D5.1 [3] and D5.2 [4].

Use Case		Goal	Machine Learning Algorithms/Frameworks	Reference
#1	Network Anomaly Detection	Detect anomalous flows in real-time through online monitoring and traffic analysis and offer an user interface that enable network administrators a deep traffic analysis.	Online real-time unsupervised network anomaly detection algorithm (ORUNADA)	D4.2, section 5

Table 1: ONTIC Use Case #1

UC #1 aims at protecting network resources and applications from malicious attacks. This problem is of key for CSPs given the continuous increase of Internet traffic and the vast expansion of networks of smart mobile devices of any kind. Evidently, these factors amplify opportunities and combinations for launching well-hidden attacks at wide range. This use case presents a typical field for applying innovative ML algorithms. It is only that such algorithms can analyse the wealth of information hidden in network traffic traces and see how traffic information evolves over time, thus enabling the identification of potential anomalous flows.

It is also important to stress that UC #1 is an actual implementation at network operation level of the algorithms developed in the scientific work packages. By combining ML and telecom expertise, this use case provides for a comprehensive network supervision interface fitting current network administration practices. At the same time, it allows for asserting network anomalies spotted by the intelligent ML detection systems by checking their validity against actual network performance.

The following points are worth mentioning.

The problems addressed by the use cases are of vital importance to CSP's especially nowadays where we witness an explosion of mobile devices and data-demanding services and applications, indicatively it is possible to mention IoT applications. UC #1 in particular aims at protecting resources and applications from malicious attacks.

The application of innovative ML algorithms for improving the performance of core network operations is currently gaining momentum. Although ML algorithms for network traffic classification are an active research topic, their integration in closed control loops with the

available network/service management systems in CSP's is generally missing. Existing control systems rely on aggregated metrics (totals, averages, min/max) and, as such, they do not exploit the wealth of evolving structural information that could be extracted from analysing raw monitored data based on ML techniques. The ONTIC use cases pave the way in this direction; their practical deployment in CSP's has been discussed in the architectural deliverable, D2.3 [5]. Note that the increasing adoption of Big Data technologies by CSP's, even as an alternative data warehouse, facilitates the deployment of the ONTIC algorithms.

Last but not least, the Use Case #1 combine ML and telecoms expertise, which is well represented in the ONTIC consortium by the mix of academic and industrial partners, respectively. Such a combination is outmost essential since it is commonly recognized that the application of ML algorithms in specific domains needs to utilize intimate knowledge of the domain itself. ML algorithms assume a generic, domain-agnostic, input model –a space of points with attributes- which obviously needs to be customized to specific application needs. This customization becomes even crucial for the application of ML traffic analysis algorithms in CSP domains since yielded analytics may trigger actions that impact on network performance, quality of the offered services and customer experience.

7.2 Use Case #1 - Network Anomaly Detection Context and Scope

UC #1 aims at designing a system able to perform online monitoring and analysis of network traffic for detecting in real-time network anomalies. As already described in deliverable D5.1 [3], the related literature refers to two kinds of ML approaches for anomaly detection: The first one leverages previously acquired knowledge as signatures or statistical models drawn from supervised learning-based approaches. The second one does not consider any acquired knowledge or training stage for initiating and configuring the detection system and its constituting algorithms. All knowledge is produced online by monitoring and analysing network traffic. Unsupervised learning algorithms are well fitted for the current fast-pacing Internet environment as already outlined previously and explained further below.

The context and objectives of UC #1 as explained in D5.1 can be summarized as follows:

- Anomalies (including attacks) are a moving target, as new anomalies and attacks arise every day. Network traffic is also constantly evolving with new applications and services appearing very frequently. The detection of new unknown anomalies (called 0d anomalies) in this changing environment is essential, and an objective of the ONTIC project. The signature- and supervised learning-based approaches cannot fulfil these requirements, since signatures and traffic statistical models have to be humanly produced, in an offline way, thus leading to long delays and cost. In addition, supervised learning approaches require training before the detection phase, which in turn requires labelled traffic traces, containing labels for all applications and anomalies that need to be detected. Evidently, building labelled traces is a very time consuming task, while it is prone to errors that can impact on the performance of the detection system afterwards.
- Traffic needs to be autonomously characterized and classified (as much as possible) in order to autonomously make a decision concerning the treatment to apply on the isolated traffic classes (legitimate or illegitimate). Relying solely on human network administrators for deciding whether a flow is legitimate leads to very poor temporal performances, and can even be useless if the attack finishes before the administrators can cope with it - attacks, for instance, are generally triggered at night, during days off, when very popular events arise, etc. that is, when network administrators are not supposed to be at work.

- Network administrators need tools to detect and analyse anomalies. Such kind of tools will provide a graphical interface and be able to work in both online (real-time) and off-line (forensic) modes. At the same time, it should be able to provide detailed information about network traffic, detected anomalies, and the relationships between them.

Given the presented context and objectives, unsupervised learning is the only promising approach. UC #1 thus leverages the online unsupervised learning algorithms based on clustering developed in WP4 for building a system able to detect anomalies (including 0d ones) and apply countermeasures in real-time, autonomously, without relying on human network administrators, previously labelled traffic traces for training, or pre-determined anomaly signatures.

The system developed in UC #1 is practically needed by network administrators; they require a tool able to display traffic monitoring results, as well as able to detect anomalies, the strongest need being related to Denial of Service (DoS) attacks. Many commercial tools exist for that purpose, but they generally lack efficiency in terms of anomaly detection; they leverage very poor first order statistics that are absolutely not suited in the context of highly variable and versatile traffic nature. As a result, their ability to detect DoS attacks is rather limited leading to high false positive and false negative rates. For instance, this is the case of the recent AlienVault solution[1], which aims at providing unified security monitoring, security events management and reporting and continuous threat intelligence as well as multiple security functions. However, it lacks real-time features and does not work in an autonomous way. Thus, it keeps on leaving most of the work to the network administrator and shows a limited usefulness.

With its ONTIC real-time unsupervised network anomaly detection algorithm, the UC #1 system is able to fix the flaws of tools such as AlienVault. It does so by providing a fully real-time, scalable and autonomous monitoring and anomaly detection tool, able to autonomously trigger counter-measures for security purposes. It is described in the reminder of this deliverable.

[1] https://www.alienvault.com/products
Indeed, only the demo version of the tool was tested and evaluated, as the price of the tool was not affordable.

8. Use Case #1 Specification

8.1 Use Cases, epics and user stories

In this section we provide a detailed description of the scenario addressed by ONTIC for the Use Case #1 (Network Anomaly Detection) by means of user stories.

In accordance with the Agile methodology, the use cases have been turned into so-called epics (high level user stories) and subsequently into user stories. The user stories have been refined throughout the project development and this document contains the final version of them.

Use Case (ONTIC DoW)	Epic (as translated in project execution time)	User Stories (as working items)
UC #1 - Network Anomaly Detection	User Story 1 (UC #1): **As a CSP or ISP network administrator**, **I want** an autonomous method for detecting analysing and characterizing traffic anomalies, **so that** it makes it possible to autonomously and efficiently manage them.	US 1.1 **As a CSP or ISP network administrator**, **I want** a mining mechanism, **so that** traffic classes can be autonomously distinguished.
		US 1.2 **As a CSP or ISP network administrator**, **I want** a discrimination mechanism **so that** anomaly signatures can be autonomously issued.
		US 1.3 **As a CSP or ISP network administrator**, **I want** a ranking score for assessing the abnormality and dangerousness of anomalies, **so that** an autonomous process can discard false attacks and chase legitimate anomalies.
		US 1.4 **As a CSP or ISP network administrator, I want** to have monitoring and analysis tools and exchange formats, **so that** the results from traffic monitoring and anomaly detection algorithms can be displayed and analysed both live and retrospectively.

Table 2: Use Case #1 (DoW) – Epics – User Stories correlation

8.2 UC #1 (User Story 1): Network Anomaly Detection

8.2.1 Scenario description

Network anomaly detection is a vital component that must exist in any network in today's Internet. Ranging from non-malicious unexpected events such as flash-crowds and failures, to network attacks such as denials-of-service and network scans, network traffic anomalies can have serious detrimental effects on the performance and integrity of the networks. The principal challenge in automatically detecting and characterizing traffic anomalies is that these are moving targets. It is difficult to precisely and permanently define the set of possible anomalies that may arise, especially in the case of network attacks, because new attacks as well as new variants of already

known attacks are continuously emerging. A general anomaly detection system should therefore be able to detect a wide range of anomalies with diverse structures, using the least amount of previous knowledge and information, ideally none.

ONTIC UC #1 designed a new autonomous anomaly detection system based on original unsupervised machine learning algorithms designed for that purpose. The most important feature of the anomaly detector is that it does not rely on previously acquired knowledge nor it needs any training phase or labelled data, while it is expected not to leverage in most of the cases on human operators for making decisions on the status of detected anomalies (legitimate vs. attack or intrusion for instance). It aims also at triggering the appropriate counter-measures in most cases.

However, as the project research results in WP4 highlight, it is not possible for the anomaly detection to autonomously make decisions for every anomaly. Therefore, a tool for a human administrator to make a decision on whether a spotted anomaly is legitimate or not; and, subsequently apply the suited counter-measure is needed. The new functionality that is required, and has been added in the design of the new anomaly detection system, is a network traffic analytics dashboard. It aims at providing human administrators with the appropriate information elements from the detection algorithms to decide what to do. The dashboard provides two sets of information:

- Legacy monitoring information on the flowing traffic.
- The characteristics of the detected anomalies as determined by the employed autonomous traffic clustering algorithm, as well as traffic statistics associated to the period during which the anomalies have been detected.

In any case, the range of possible anomalies is too large and sometimes the anomalies automatically detected are unclear or even unknown. Therefore, network administrators need an information system that enable them to know the status of the network at any time and also to detect network incidents. With the help of such an information system, they could perform detailed analysis and make quick and effective decisions. The most appropriate solution to meet this need is an application in the form of a network traffic analytics dashboard. This application should provide detailed information on both network traffic and anomaly detection as well as enable visualization and analysis, both in real time (on-line) for making decisions timely and in "forensic" (off-line) mode for allowing further analysis.

8.2.2 User Requirements

The functional specification for UC #1 is described as a set of user stories exposed below:

- **As a CSP or ISP network administrator, I want** an autonomous method for detecting and characterizing traffic anomalies, **so that** it makes it possible to autonomously and efficiently manage them.
    - User Story 1.1: **As a CSP or ISP network administrator, I want** a mining mechanism, **so that** traffic classes can be autonomously distinguished.
        - User Story 1.1.1: **As a CSP or ISP network administrator, I want to** have efficient monitoring, unsupervised clustering techniques and related analytics, **so that** I can autonomously classify network traffic. →Implementation done

- User Story 1.2: **As a CSP or ISP network administrator, I want** a discrimination mechanism, **so that** anomaly signatures can be autonomously issued.
  - User Story 1.2.1: **As a CSP or ISP network administrator, I want to** have mechanisms for identifying the most significant traffic attributes, **so that** it becomes possible to issue traffic class discrimination rules. → Implementation done
- User Story 1.3: **As a CSP or ISP network administrator, I want** a ranking score for assessing the abnormality and dangerousness of anomalies, **so that** an autonomous process can discard false attacks and chase legitimate anomalies
  - User Story 1.3.1: **As a CSP or ISP network administrator, I want to** have accurate abnormality scores, **so that** it becomes possible to autonomously discriminate between legitimate and illegitimate traffic classes. → Implementation done
- User Story 1.4: **As a CSP or ISP network administrator, I want** to have monitoring and analysis tools and exchange formats, **so that** the results from traffic monitoring and anomaly detection algorithms can be displayed and analysed both live and retrospectively
  - User Story 1.4.1: **As a CSP or ISP network administrator**, **I want to** have a data network traffic analytics dashboard to show traffic and flow statistics, anomaly detection details, etc., **so that** I will be able to analyse data traffic features and study in depth the anomalies detected.
    - User Story 1.4.1.1: **As a CSP or ISP network administrator**, **I want** to get traffic analysis charts, **so that** I can have a well-aimed knowledge about the state of the network.
      - » User Story 1.4.1.1.1: **As a CSP or ISP network administrator**, **I want** to get a Real-Time traffic analytics visualization tool, **so that** I can view overall traffic statistics regarding IPs, ports, type of service, bytes, etc.

        → Implementation finished.
      - » User Story 1.4.1.1.2: **As a CSP or ISP network administrator**, **I want** to get a real-time flow analysis tool, **so that** I can view precise statistics related to traffic flows, such as conversations. → Implementation finished.
      - » User Story 1.4.1.1.3: **As a CSP or ISP network administrator**, **I want** the anomaly detection tool to show a warning message whenever an anomaly has been detected, **so that** I can become aware of potential abnormal situations any time they happen and obtain further information by accessing the tool.

        → Implementation finished
      - » User Story 1.4.1.1.4: **As a CSP or ISP network administrator, I want** to be able to specify the time window of the traffic analysis by choosing a specific number of minutes back from current time, **so that** I have a flexible way for seeing the network situation and the evolution of traffic in short term intervals and for getting further details regarding the spotted anomaly and any relevant event instantly.

        → Implementation finished

» User Story 1.4.1.1.5: **AS a CSP or ISP network administrator**, **I want** to be able to select a specific date and a time period to see all network information available (traffic and anomalies) both in dynamic and in static modes, **so that** I can make in-depth forensic analysis based on historical data.

→ Implementation finished.

- User Story 1.4.1.2: **As a CSP or ISP network administrator**, **I want** to get an anomaly detection tool, **so that** whenever a traffic anomaly is detected I will be aware of it at once along with its details.
  → Implementation finished
- User Story 1.4.1.3: **As a CSP or ISP network administrator**, **I want** to have a set of administration procedures, **so that** it is possible to manage and configure different system features.

  → Implementation finished.
- User Story 1.4.1.4: **As a CSP or ISP network administrator**, **I want** to have a login/password authentication procedure, **so that** it is possible to prevent unauthorized parties from accessing the anomaly detection tool.
  → Implementation finished
- User Story 1.4.1.5: **As a CSP or ISP network administrator, I want** to have a comprehensive system view to be always displayed in a shared screen (a video-wall for instance), **so that** the whole team of network administrators can have a general view about the network status and be warned of any incidence in real time.

  → Implementation finished.
- User Story 1.4.1.6: **As a CSP or ISP network administrator**, **I want** to be automatically informed when a new anomaly is detected by the system, **so that** I can reduce the time to check the anomaly and manage the alert properly.

  → Implementation finished.
- User Story 1.4.1.7: **As a CSP or ISP network administrator**, **I want** to generate technical reports with the dashboard information, **so that** I can select a specific time period and generate an electronic document including traffic information and all anomalies detected during that period.

  → Implementation finished.
- User Story 1.4.1.8: **As a CSP or ISP network administrator**, **I want** to export stored data in the system in a well-known interchangeable file and data-type formats, **so that** I can further analyse dashboard data with other more powerful analytics tools.

  → Implementation finished.

8.2.3 Requirement Specification

8.2.3.1 Anomaly Detection Subsystem Requirement Specification

The Anomaly Detection Subsystem is based in the implementation of ORUNADA algorithm described in the last version of the ONTIC deliverable D4.2 [1] provided in June 2016. Note that a new version of ORUNADA is being designed and developed nowadays, and it is described in

deliverable D4.3 [2]. This new version is due to the fruitful collaboration between ONTIC and the H2020 ENDEAVOUR project. IBM, one of the ENDEAVOUR partners, requires efficient anomaly detection tools, with very strong throughput constraints, i.e. much larger than the ones that were targeted in ONTIC Use Case #1. D4.3 [2] presents the implementation of this new ORUNADA version and its first performance evaluation results. In this deliverable, the target is to cope with the traffic of the SATEC datacentre. Given the traffic constraints, as well as the ONTIC schedule, the version that has been developed for Use Case #1 is the one described in deliverable D4.2 [1].

Four Use Case #1, the main requirements of ORUNADA and its implementation are the following:

- Detecting and characterizing anomalies from traffic captured at the PCAP format.
- Generating reports with the characteristics of the detected anomalies. Reports are coded in XML format.
- Detecting anomalies and providing reports in online mode.
- Executing in near real-time.

8.2.3.2 Dashboard Subsystem Requirement Specification

The requirement specification for the Dashboard subsystem corresponds to user story 1.4 and has been expressed with the UML (Unified Modeling Language) Use Case model shown in Figure 1.

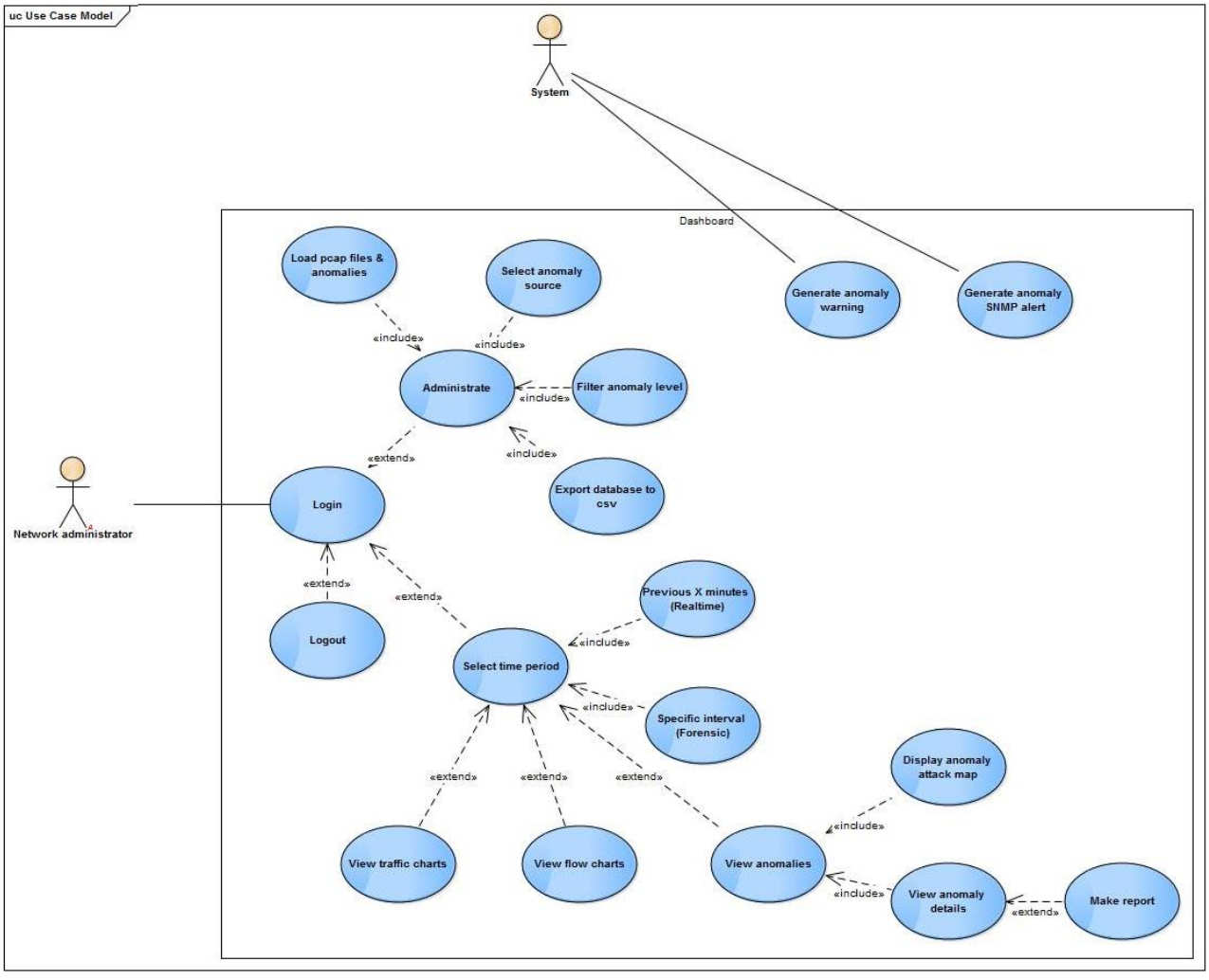


Figure 1: Dashboard UML Use Case Model.

The model is driven by to actors: “Network Administrator” (as the user of the subsystem) and “System” (as the internal control).

The Use Case descriptions are defined in the following tables:

Use case ID	UC#1-DB1
Use Case Name	Login
Description	Network Administrator enters the system by means of a password
Primary actor	Network Administrator
Preconditions	N/A
Post-conditions	The administrator is logged into the system
Basic flow	1. Administrator enters his/her ‘user’ and ‘password’ into the login form 2. Administrator clicks on the ‘sign in’ button 3. Administrator enters the dashboard
Alternate flow	3a. ‘user’ and/or ‘password’ are wrong, so the administrator is asked to introduce them again in the form

Table 3: Use Case UC#1-DB1.

Use case ID	UC#1-2
Use Case Name	Load PCAP file and anomalies
Description	Network administrator enters the admin section and chooses a PCAP file to analyse it and load in the database.
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the admin panel
Post-conditions	The chosen PCAP file will start being loaded in the database. This process takes a few minutes to complete
Basic flow	1. Administrator clicks "load Pcap" button  located in the lower right corner. 2. Administrator selects the PCAP file from the select box item 3. Administrator clicks on the ‘load PCAP’ button 4. System converts PCAP records into NetFlow records and send it to the process pipeline (filter, enrich, analyse, store, …)
Alternate flow	N/A

Table 4: Use Case UC#1-DB2.

Use case ID	UC#1-3
Use Case Name	Select anomaly source
Description	Network administrator enters the *settings* section (menu located on the upper right corner) and selects any of the anomaly sources available
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the admin panel
Post-conditions	The chosen option will be used in the dashboard as the anomaly source
Basic flow	1. Administrator clicks the settings section in the menu 2. Administrator selects the anomaly source from the select box item 3. Administrator clicks on the 'Change' button 4. The selected source will be used in the dashboard to read anomalies from.
Alternate flow	N/A

Table 5: Use Case UC#1-DB3.

Use case ID	UC#1-4
Use Case Name	Filter anomaly level
Description	Network administrator enters the *settings* section (menu located on the upper right corner) and selects an anomaly critical level from a relevant menu in order to filter the anomalies to display
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the admin panel
Post-conditions	The chosen option will be used in the dashboard to filter anomaly per level of importance
Basic flow	1. Administrator clicks the settings section in the menu 2. Administrator selects the anomaly level from the select box item 3. Administrator clicks on the 'Change' button 4. The selected level will be used in the dashboard to filter the anomalies to show.
Alternate flow	N/A

Table 6: Use Case UC#1-DB4.

Use case ID	UC#1-5
Use Case Name	Export database to CSV
Description	Network administrator enters the *Export* section (menu located on the left side); selects type of data records to be exported (anomalies or NetFlows) and selects a period of time that wishes to download data for (as CSV file).
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the admin panel
Post-conditions	N/A
Basic flow	1. Administrator clicks the *Export* section in the menu on the left. 2. Administrator selects the type of data to be exported (NetFlow or anomalies) from the select box item. 3. Administrator selects the period using the calendar boxes 4. Administrator clicks on the 'Download' button 5. The generated CSV file will begin downloading
Alternate flow	N/A

Table 7: Use case UC#1-DB5.

Use case ID	UC#1-6
Use case Name	Previous X minutes (Real-time)
Description	Network administrator selects the last 5, 15, 30, … minutes (counting from the current moment) to display data in the dashboard
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	The dashboard is set to Real-time mode, displaying data that is being read at the moment from the network link.
Basic flow	1. Administrator selects real time mode clicking *RT* button located on the lower right corner. 2. Administrator clicks the desired option from the menu "select last minutes": '5 min', '15 min', '30 min', etc. 3. Data corresponding to the last X minutes will be displayed.
Alternate flow	N/A

Table 8: Use Case UC#1-DB6.

Use case ID	UC#1-7
Use Case Name	Specific time interval (for Forensics)
Description	Network administrator selects the specific period to display data in the dashboard.
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	The dashboard is set to forensic mode, displaying past data that is stored in the database
Basic flow	1. Administrator clicks the 'Forensic' option (FO button) from the menu located on the lower right corner. 2. Administrator selects the period using the calendar boxes. 3. Administrator chooses the option (on, off) to configure a minimum interval to display. 4. Administrator clicks on the play button to start the simulation 5. Data corresponding to the desired periods will be displayed, refreshing every X seconds, till the 100% of the period is covered 6. Administrator can pause, go forward/backward or stop the simulation at any moment. 7. Administrator can select the simulation speed (x1, x2, x4 or x8)
Alternate flow	2a. The beginning of the period is a later date than the end, then the calendar boxes are cleared and the administrator is asked again to set a period 3a. When minimum interval has been set to on the administrator sets in the input bar the number of seconds that the charts will be refreshing over and over.

Table 9: Use Case UC#1-DB7.

Use case ID	UC#1-8
Use Case Name	View traffic charts
Description	Network administrator visualizes charts related to network traffic
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	N/A
Basic flow	1. Administrator clicks the 'Traffic analysis' option from the menu located on the left side. 2. The traffic charts panel will be displayed
Alternate flow	N/A

Table 10: Use Case UC#1-DB8.

Use case ID	UC#1-9
Use Case Name	View flow charts
Description	Network administrator visualizes charts related to network traffic at flow level, as conversations.
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	N/A
Basic flow	1. Administrator clicks the 'Flow analysis' option from the menu located on the left side. 2. The flow charts panel will be displayed
Alternate flow	N/A

Table 11: Use Case UC#1-DB9.

Use case ID	UC#1-10
Use Case Name	View anomalies
Description	Network administrator visualizes the list of all the anomalies detected
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	N/A
Basic flow	1. Administrator clicks the 'Anomalies' option from the menu located on the left side. 2. The anomalies panel will be displayed
Alternate flow	N/A

Table 12: Use Case UC#1-DB10.

Use case ID	UC#1-11
Use Case Name	Display anomaly attack map
Description	Network administrator keeps track in real-time of the attacks happening, geo-localized in a world map, in a separate window
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the anomalies panel
Post-conditions	N/A
Basic flow	1. Administrator clicks the 'Attack map' button located on the upper right corner. 2. The attack map window is prompted
Alternate flow	N/A

Table 13: Use Case UC#1-DB11.

Use case ID	UC#1-12
Use Case Name	View anomaly details
Description	Network administrator digs into the details of an anomaly, i.e. examines traffic charts during the anomaly's period
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system and in the anomalies panel
Post-conditions	N/A
Basic flow	1. Administrator clicks the anomaly from the list. 2. The anomaly details are prompted
Alternate flow	N/A

Table 14: Use Case UC#1-DB12.

Use case ID	UC#1-13
Use Case Name	Make report
Description	Network administrator generates a report with the details of an anomaly
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system, in the anomalies panel and in the details of an anomaly
Post-conditions	N/A
Basic flow	1. Administrator clicks the anomaly from the list (Anomalies view). 2. Administrator clicks the 'Report (pdf)' button. 3. The report with the details of the anomaly is downloaded as pdf.
Alternate flow	N/A

Table 15: Use Case UC#1-DB13.

Use case ID	UC#1-14
Use Case Name	Generate anomaly warning
Description	The dashboard displays a popup for every newly detected anomaly
Primary actor	The system
Preconditions	The administrator must be logged into the system
Post-conditions	N/A
Basic flow	1. The dashboard shows a warning popup.
Alternate flow	1a. Administrator can click on the warning to go to the anomalies panel

Table 16: Use Case UC#1-DB14.

Use case ID	UC#1-15
Use Case Name	Generate anomaly SNMP alert
Description	The system sends a SNMP trap every time a new anomaly is detected to the SNMP client which in turn might send an email to the network administrator, for instance
Primary actor	The system
Preconditions	N/A
Post-conditions	N/A
Basic flow	1. The system sends a trap to the SNMP client if a new anomaly is detected
Alternate flow	1a. The SNMP client could send an email to the administrator

Table 17: Use Case UC#1-DB15.

Use case ID	UC#1-16
Use Case Name	Logout
Description	Network administrator leaves the dashboard
Primary actor	Network administrator
Preconditions	The administrator must be logged into the system
Post-conditions	The administrator is logged out the system
Basic flow	1. Administrator clicks 'Logout' option in the menu located on the upper right corner
Alternate flow	

Table 18: Use Case UC#1-DB16.

9. Use Case #1 Design

9.1 System model

9.1.1 Functionalities

Based on the specification of the user stories US 1.1 through US 1.4, two main system functions are provided: (a) an autonomous system for detecting and characterizing traffic anomalies, making it possible to autonomously and efficiently manage them: the **Anomaly Detection Subsystem**, and (b) a dashboard for enabling network operators to get detailed information about network traffic features and statistics, near real-time, anomalies detected and traffic behaviour during the periods in which the anomalies are detected: the **Network Traffic Dashboard Subsystem**.

Figure 2 represents the high level working schema of the UC #1 system. PCAP files, containing traffic traces, provide input to both subsystems –the anomaly detection and the dashboard subsystems. The results of the anomaly detection process are fed, in the form of XML files, to the dashboard for further investigation and permanent store.

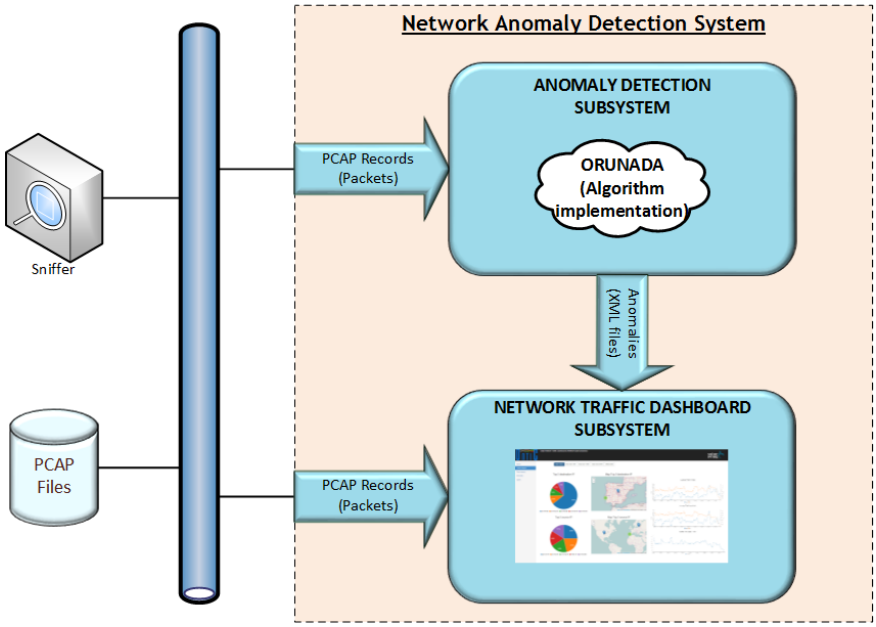


Figure 2: Use Case #1 High-level Architecture

The Anomaly Detection Subsystem analyses currently flowing network traffic that is captured at the PCAP format by applying the ORUNADA unsupervised anomaly detection algorithm, presented in deliverable D4.2 [1]. When it detects an anomaly, it sends an XML file to the Network Traffic Dashboard Subsystem. The file contains the characteristics of the detected anomaly, for display purposes and for autonomously launching counter-measures, if required.

The Network Traffic Dashboard Subsystem entails the following functionalities (Figure 3):

- Ingesting captured network traffic traces, near real-time, at different formats (PCAP and NetFlow) and from different sources, through a scalable software system that can elastically scale with the incoming traffic rate. When no NetFlow input source is available, the system translates PCAP into NetFlow and injects the resulted NetFlow streams as a new input data source. For scalability reasons, the system is designed to process traffic traces

at flow level (NetFlow) rather than at raw packet level (PCAP) since flow traces aggregate packet traces without losing information.

- Receiving data from external analysis systems, for example, analysis results about anomalies from the Anomaly Detection Subsystem (XMLs files), or other traffic analytics systems.
- Translating received raw data (PCAP, NetFlow, XMLs with anomaly analysis results, etc.) into structured records as required for processing.
- Storing all received data in an elastic data base and processing it in a scalable manner.
- Querying stored data and presenting the results in a structured way in graphical and/or in tabular/alphanumeric forms.

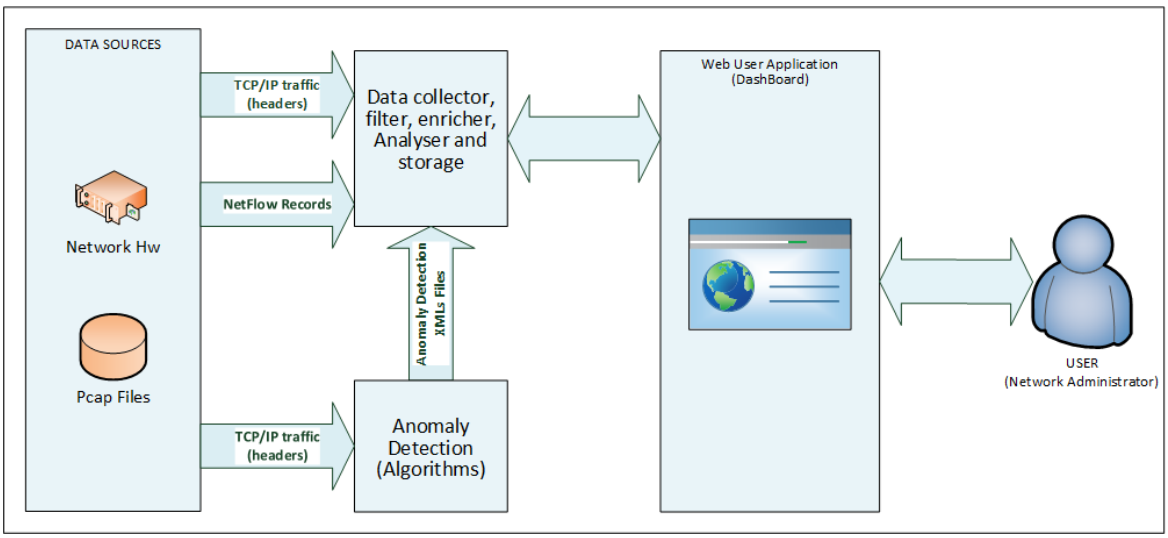


Figure 3: UC #1 Dashboard Functional View

The output of the Network Traffic Dashboard Subsystem is a user web interface (network operator oriented) shown as an analytics dashboard which presents traffic analysis results in a structured way with graphics and tabular/alphanumeric data.

9.1.2 Software architecture

The software architecture and the relationships between the different components of the UC #1 system are depicted in the UML diagram shown in Figure 4.

The diagram also uses colours to distinguish contributions from different work packages:

- Modules in red are those developed in WP4 on online clustering algorithms for anomaly detection;
- Modules in green and pink implement the Network Traffic Dashboard Subsystem; they handle pre-processing of traffic data, computation of statistics, processing of anomaly detection reports, and visualization of statistics and anomaly detection information. Green modules need the intervention for an external actor;
- Modules in grey are components not directly under the scope of this use case, but included as they can significantly improve the use case demonstration.

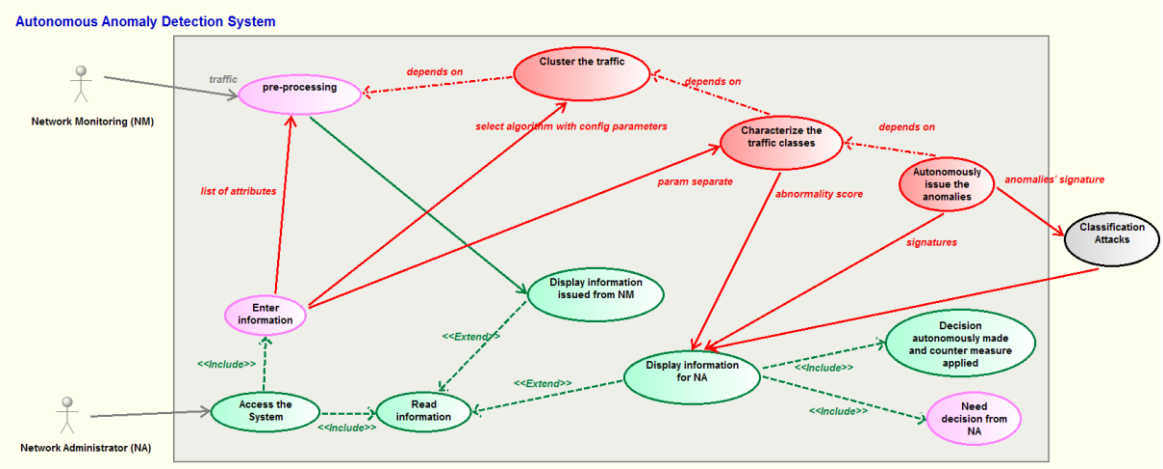


Figure 4: UML specification of UC #1 on anomaly detection

This UML diagram shows two main threads: one in charge to information processing and results presentation, and other handling anomaly detection.

- The **information processing and results presentation** thread is devoted to traffic monitoring and real-time results display in the Network Traffic Dashboard Subsystem. It comprises two modules:
    1. **Display information from NM (Network Monitoring)**: It is fed with processed traffic records (generated by the "pre-processing" module) and anomaly classification results (provided by the "classification" module) and performs the necessary computations for issuing a real-time display of selected features regarding traffic flows and detected anomalies)
    2. **Display information for NA (Network Administrator)**: It gets information from the clustering algorithms used to detect traffic anomalies And displays live in the Network Traffic Dashboard Subsystem the characteristics of the identified traffic classes, the abnormality scores, and the results of attack classification in order to help the network administrator to make a decision ("Need decision from NA" module). If the Anomaly Detection Subsystem can autonomously make the decision, the Network Traffic Dashboard Subsystem will display that a corrective action has been taken, for example, it will display a text alert saying "Decision autonomously made and counter measure applied", along with details about the measure applied.
- The **anomaly detection** thread comprises the modules implementing the core anomaly detection algorithm. Specifically, it includes three main sequential steps: clustering the traffic, issuing the characteristics of the resulted traffic classes, and autonomously issuing the anomalies when autonomous detection is possible, otherwise leveraging found anomalies' signature information ("Classification attacks" module).

9.1.3 Anomaly Detection Subsystem Design

The design and implementation of the unsupervised network anomaly detection algorithm, called ORUNADA, is presented in deliverable D4.2 [1]. Please also refer for a comprehensive description to the paper accepted for publication in IEEE Transactions on Network and System Management [15].

9.1.4 Network Traffic Dashboard subsystem Design

The general structure of the Network Traffic Dashboard Subsystem is depicted in Figure 5; it includes two modules:

- Network Data Processing
- Result Visualization

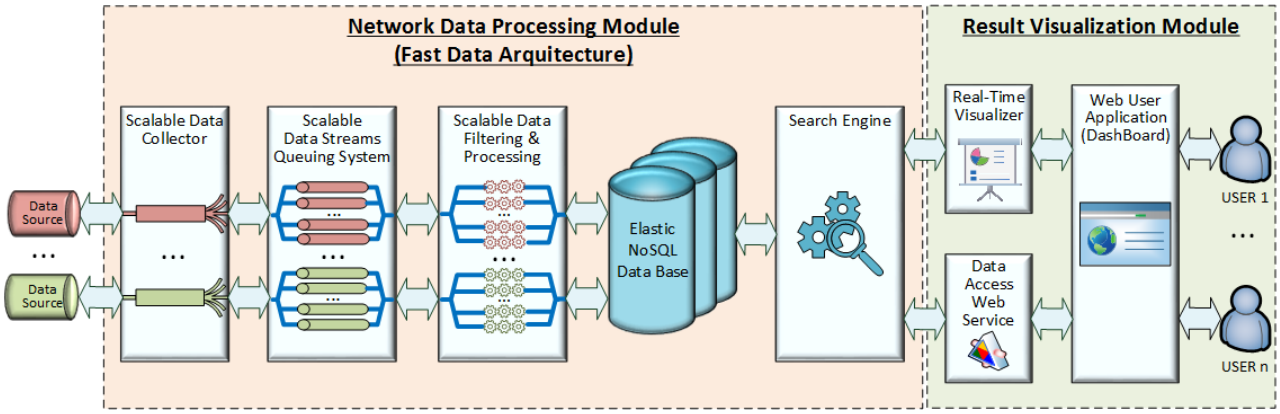


Figure 5: Network traffic and anomaly detection dashboard architecture.

The architecture of the Network Data Processing Module is based in a "Fast Data Model" as described in a paper accepted and published in the IEEE International Conference on Digital Information Management [17], and it is made up of the following components:

- A set of data traffic sources such as:
  - ONTS (ONTIC Network Traffic Summary) data set [18][19]: PCAP files with captured traffic at raw packet level.
  - Anomaly Detection Subsystem: XML files with information on the identified anomalies.
  - Network hardware: NetFlow v5 records[2].
- Network Traffic Data Collector module: scalable pool of pipelines to pre-process input data. Pipelines are commonly associated to data sources. The following pipelines are implemented:
  - PCAP pipeline: It reads PCAP files and converts TCP/IP headers to flows in NetFlow format.
  - NetFlow pipeline: It receives NetFlow records through a defined UDP port and pre-processes these records, for example, for creating tuples with new conversations, and stores the resulted data in the database.
  - Anomaly detection pipeline: It receives XML files from the Anomaly Detection Subsystem (from a commonly agreed file directory or through a Web Service implemented in the Network Traffic Dashboard Subsystem), parses the XML files (e.g. to discriminate between new and previous anomalies that continue active) and stores the results in the database.
- Scalable NoSQL Database: to store data.

[2] http://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html

- Search Engine: to provide data access and execute queries.

The main components of the Result Visualization Module are:

- Visualizer: a set of libraries to convert data to charts.
- Data Access Web Service: an API to provide a query system over stored data.

9.1.4.1 Network Data Processing Module design

The Network Data Processing module is made up of a set of distributed parallel pipelines. It is described by both its UML class diagram and associated sequence diagrams:

- The pipeline responsible for processing NetFlow records consists of the following classes (Figure 6):

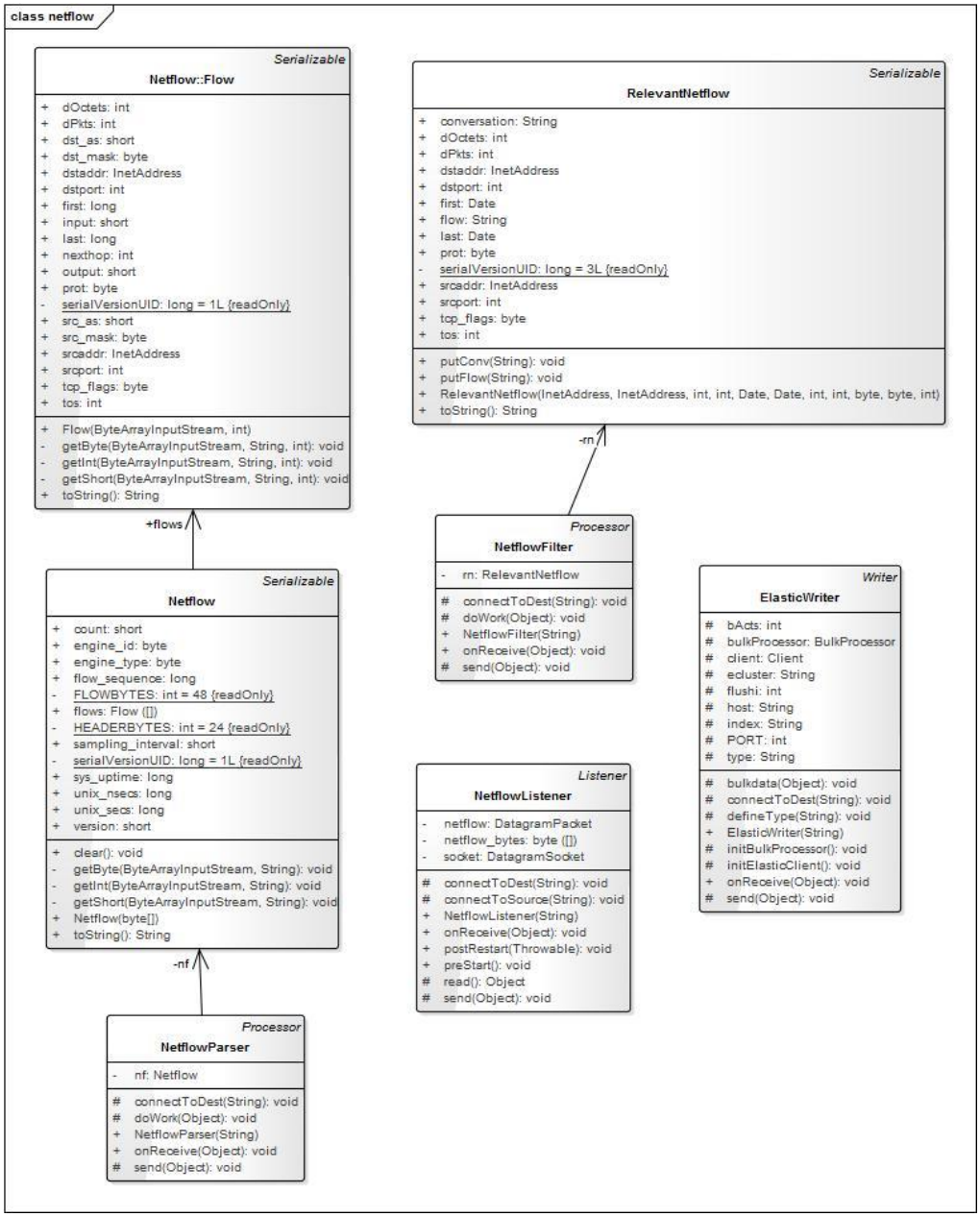


Figure 6: NetFlow classes.

- The classes behind the pipeline for processing XML anomaly files are (Figure 7):

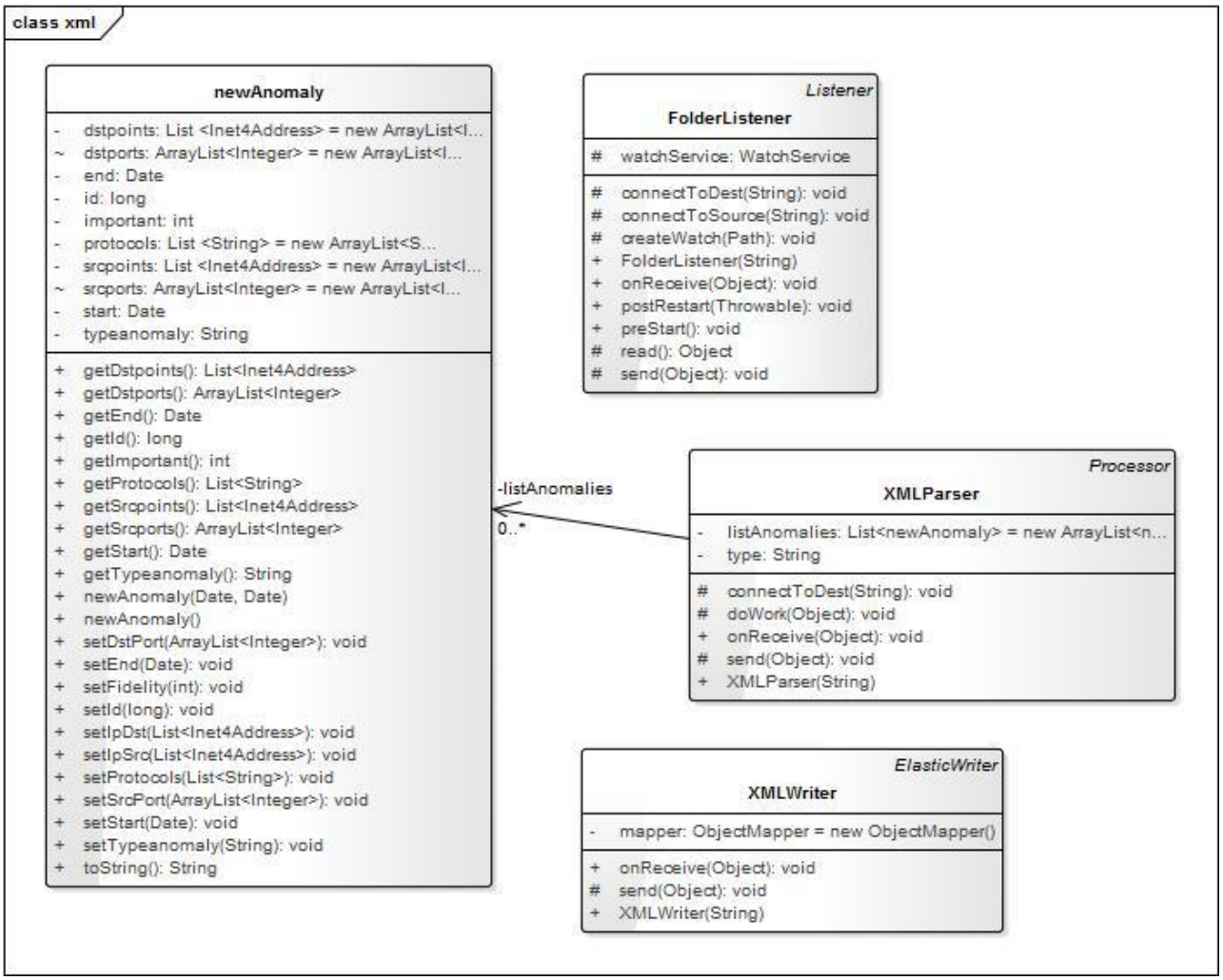


Figure 7: Anomaly XML classes.

- The way pipelines process their respective data sources is explained in the following sequence diagram (Figure 8):

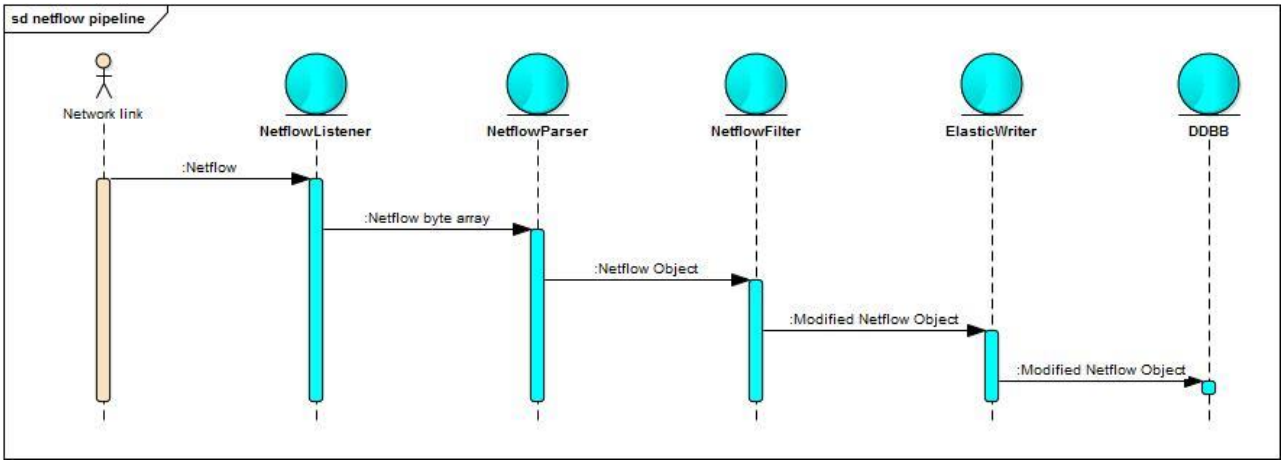


Figure 8: Get list of anomalies sequence diagram.

9.1.4.2 Result Visualization Module design

The Result Visualization module consists of a server that queries the Search Engine and a web client that converts received data into charts using a set of libraries.

The module is defined using UML class diagrams and sequence diagrams as presented in the following. The Controller classes realise the interface of the web client (Figure 9):

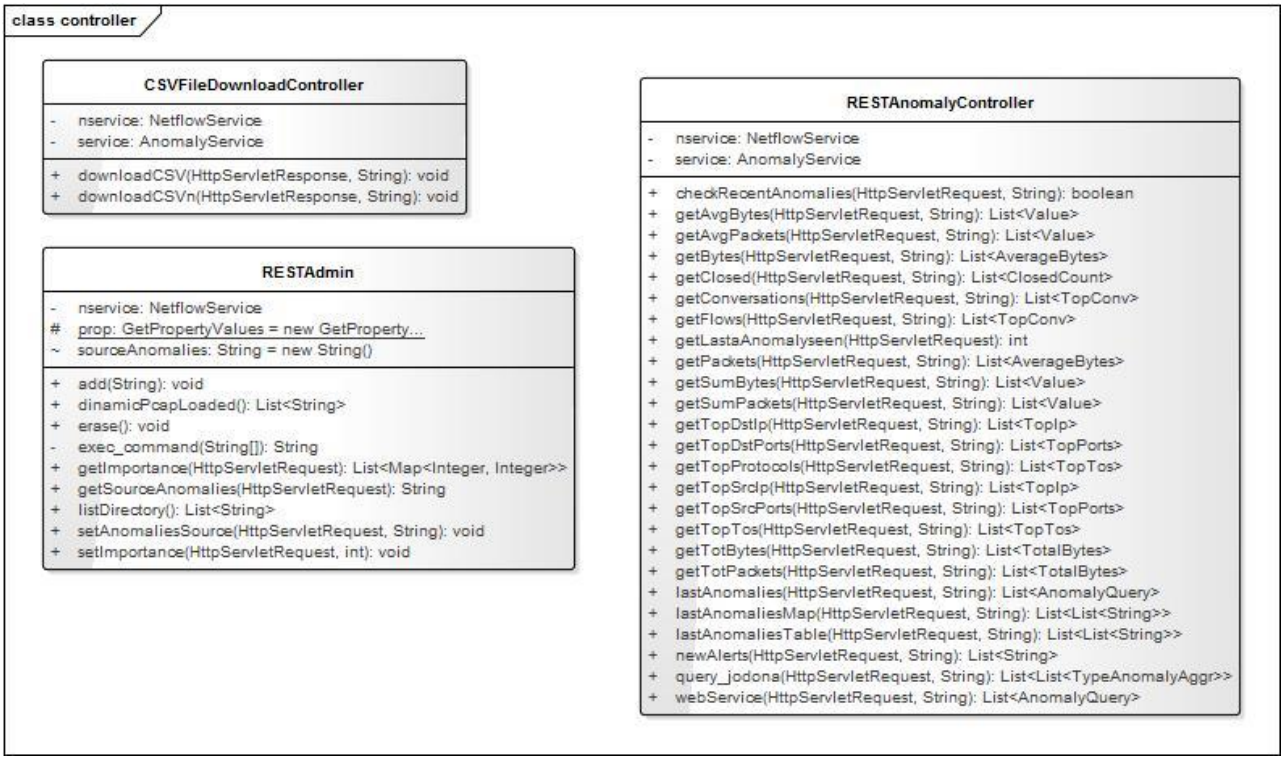


Figure 9: Controller classes.

At request from the client, Controller classes call Service classes to query data from the database (Figure 10):

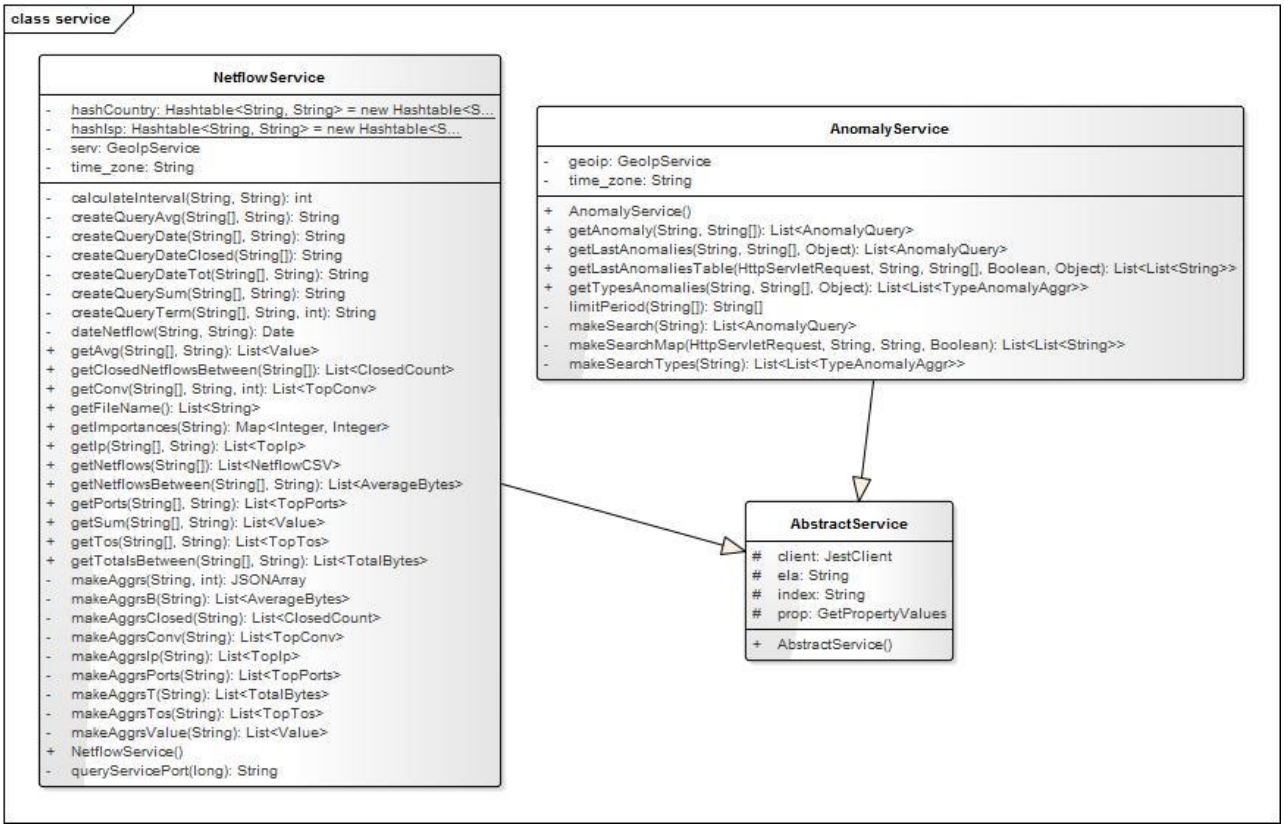


Figure 10: Service classes.

Among the many use cases related to the operation of the Result Visualization module, indicative examples include:

- Load NetFlow records in the database from a PCAP file (Figure 11):

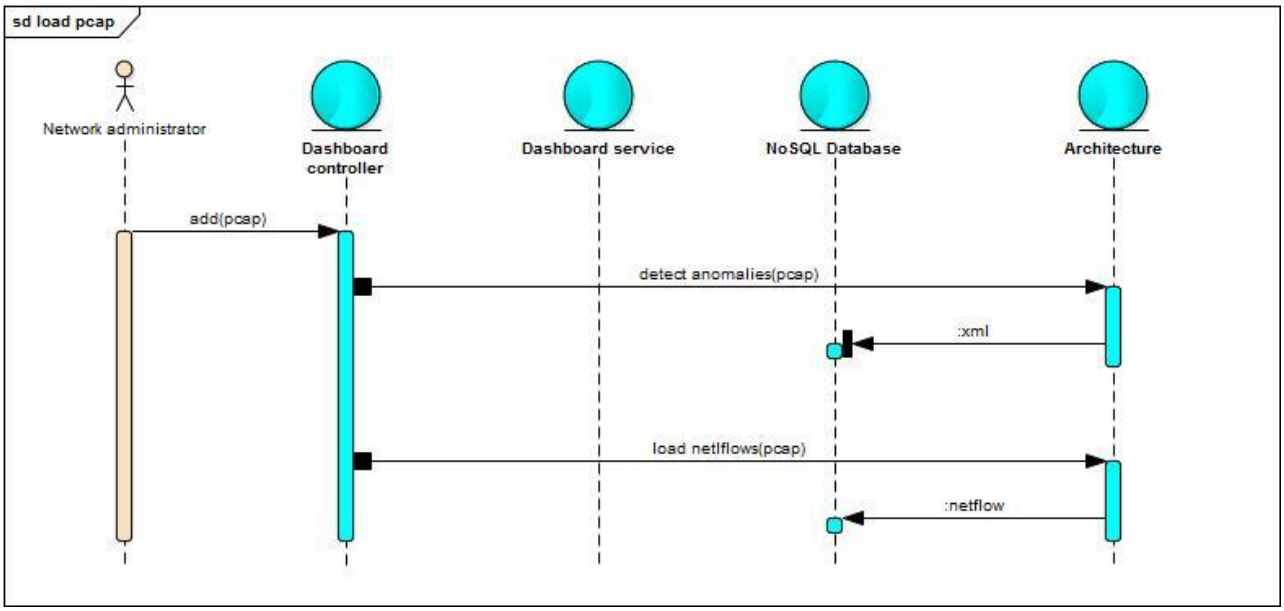


Figure 11: Load PCAP file sequence diagram.

- Dump database as CSV file (Figure 12):

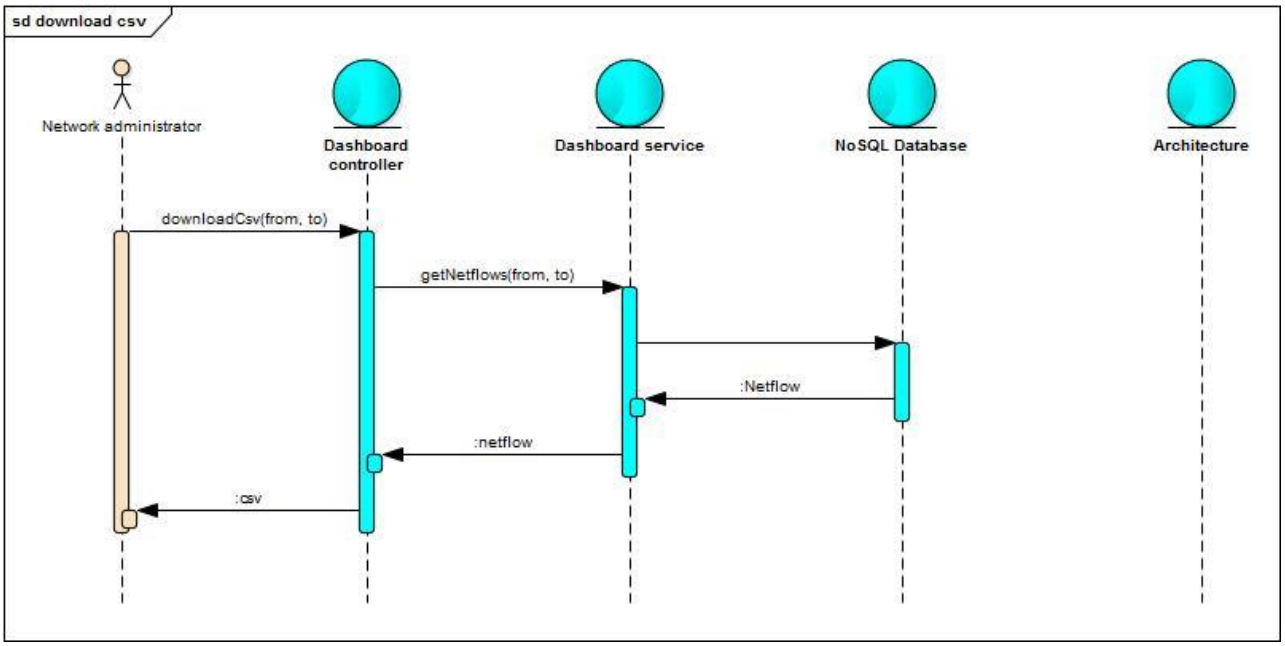


Figure 12: Download CSV sequence diagram.

- Get top 5 source IPs (Figure 13):

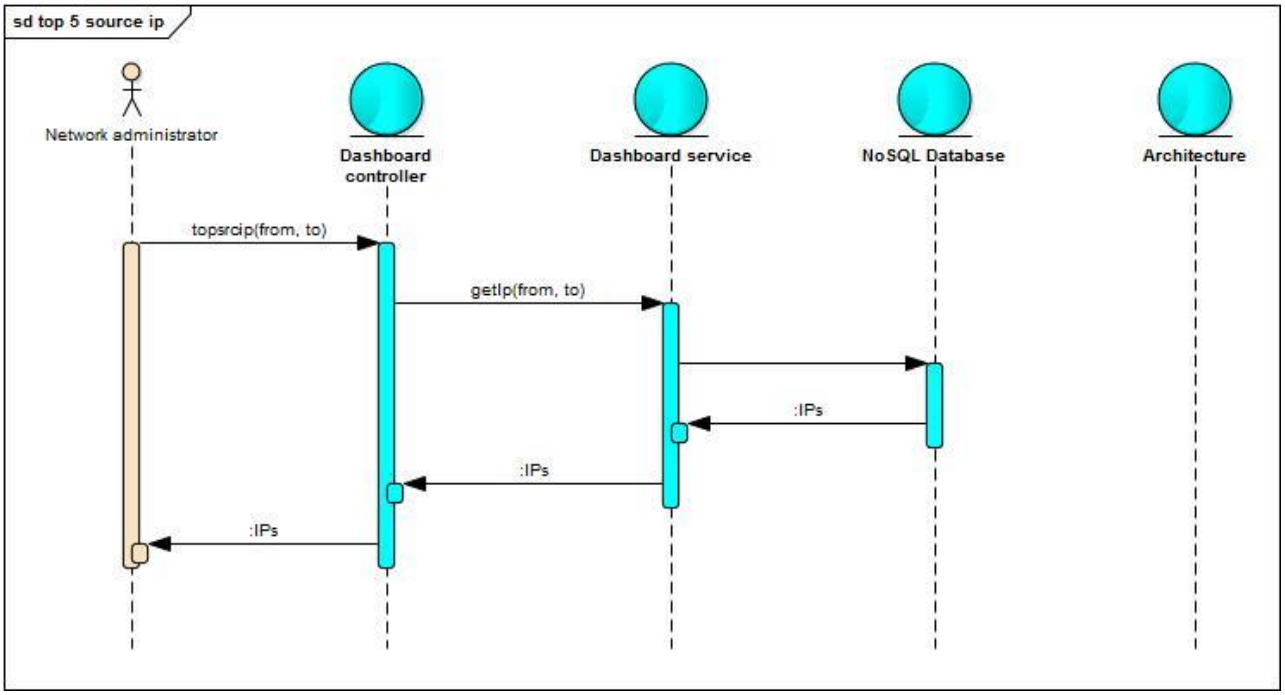


Figure 13: Get top source IP sequence diagram.

- Get top 5 conversations (Figure 14):

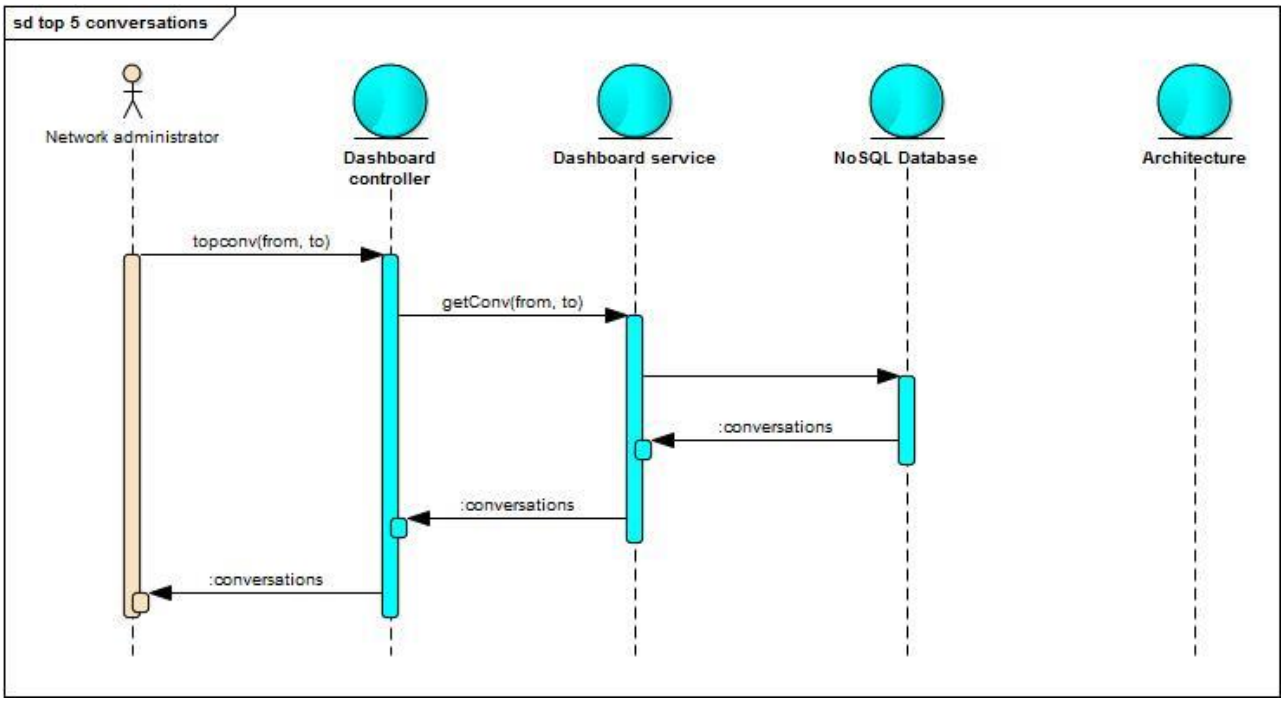


Figure 14: Get top conversations sequence diagram.

- Get list of all anomalies (Figure 15):

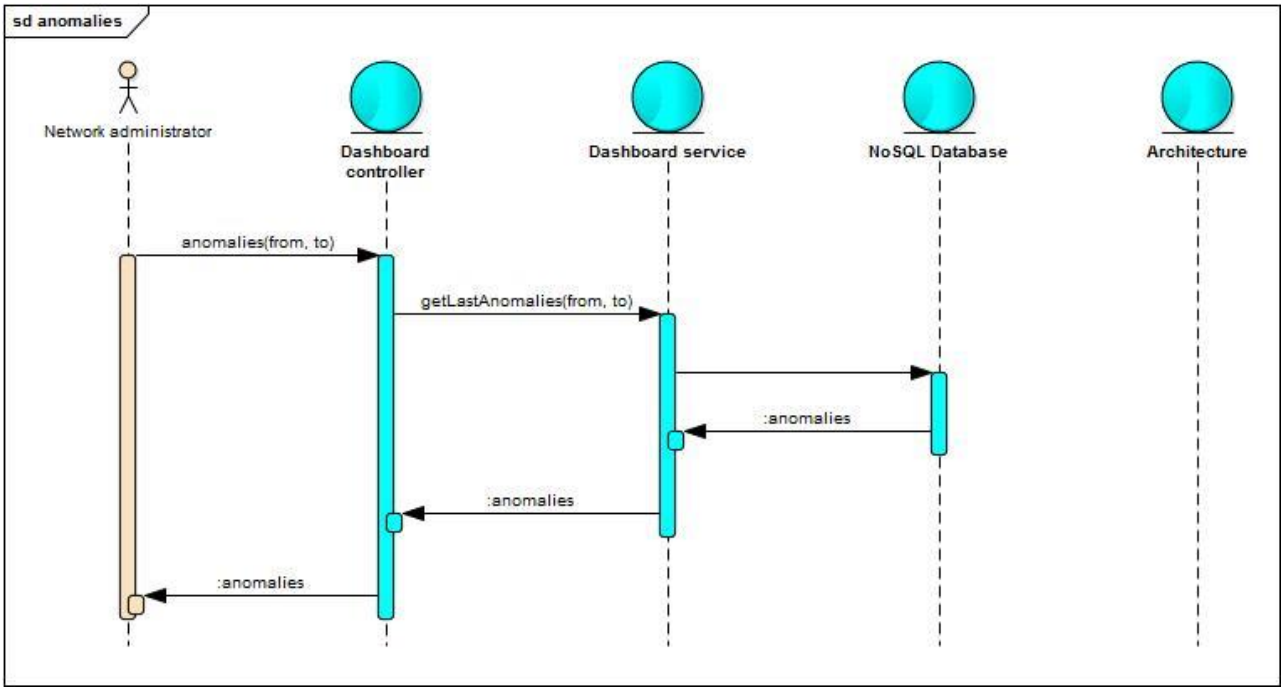


Figure 15: Get list of anomalies sequence diagram.

10. Use Case #1 Implementation

10.1.1 Anomaly Detection Subsystem implementation

As already stated, the ORUNADA algorithm developed in WP4 and described in [1] is in charge of analysing the incoming traffic and detecting anomalies. To cope with the real-time constraints for this kind of analysis, the ORUNADA algorithm takes advantage of:

- A sliding window that shifts frequently from one micro-slot to the other, for allowing enough traffic data to be passed within each window as required for accurate detection results, without delaying processing more than the duration of one micro-slot;
- IDGCA (Incremental Density Grid-based Clustering Algorithm) for fast computations in the framework of small traffic window shifts;
- An exhaustive and free of redundancy list of 18 traffic features relevant for detecting network attacks.

ORUNADA has been implemented in Java v1.8 (see code repository in https://gitlab.com/ontic-wp4/ORUNADA). ORUNADA detects anomalies based on traffic inputs supplied in PCAP files in a continuous way. It also generates signatures to describe identified anomalies.

ORUNADA outputs on the standard output information about its execution. It also creates an XML file for each micro-slot processed (apart for the n first micro-slots, n equals to the number of micro-slots in a window). This XML lists the anomalous flows found in the PCAP file at the end of each micro-slot considering the packets contained in the current window. For each anomalous flow it specifies its features, its score of dissimilarity and its signature.

10.1.2 Network Traffic Dashboard Subsystem implementation

The Network Traffic Dashboard Subsystem implements an ISP/CSP network administration-oriented tool that provides online (real-time) and offline (forensic analysis) traffic monitoring and analysis capabilities, as well as anomaly detection.

The subsystem uses captured traffic datasets (in PCAP or NetFlow v5 format) and the XML output of the Anomaly Detection Subsystem with information on identified anomalous flows. As outlined in section 9.1.4 these input sources are processed through the Network Data Processing module in a series of four functional stages: ingestion, parsing, filtering and storage. Finally, the Result Visualization module queries the stored data to display analytics in real-time on a web interface.

Such a scenario requires the Network Data Processing and Result Visualisation modules to support a series of operational characteristics related to performance and scalability requirements in order to cope with the challenges pertinent in realistic deployment, i.e. monitoring and analysing traffic from real network links. Such characteristics include: sustain high volumes of input data without losses, engage parallel processing of all input streams in a distributed, scalable and highly available manner in order to achieve near real-time performance. Hence, the Network Data Processing module must be able to handle increased volumes of work load, which means it should be able to optimise its performance accordingly by engaging more/less resources (scale-up/down).

A library like Akka comes in handy for such expectations. Akka is a Java library that relies upon the so-called Actor Model to build applications. This model offers features such as:

- Simple and high-level abstractions for distribution, concurrency and parallelism.

- Asynchronous, non-blocking and highly performant message-driven programming model that ensures loose coupling, isolation and location transparency.
- Very lightweight, event-driven processes (several million actors per GB of heap memory).
- Responsive systems; Akka focuses on providing rapid and consistent response times so that it delivers a stable quality of service.
- Highly-available and fault tolerant support systems, isolating components from each other and thereby ensuring that parts can fail and recover without compromising the system as a whole ("let it crash" model).
- Elasticity or responsiveness under varying workloads. This is achieved by increasing or decreasing the processing, memory and disk resources allocated on commodity hardware by, for example, replicating components and distributing inputs among them.

Consequently, the Network Data Processing module was built on top of Akka library.

The point of convergence between the two modules comprising the Network Traffic Dashboard Subsystem is the database where data is stored and queried. Due to the need of horizontal scaling to clusters of machines, which is a problem for relational databases, plus the heterogeneity of the data structures used, and a bunch of fast operations required for achieving ear real-time processing (like complex searches), a NoSQL database suits better than a typical relational database.

Elasticsearch is a highly scalable full-text search and analytics engine. It allows to store, search, and analyse big volumes of data quickly and in near real-time. Thus, it can fit the requirements of the underlying engine/technology for powering the dashboard subsystem's needs for storage and analysis.

Taking into account that the prototype environment does not provide a NetFlow input data source, the decision to integrate a module to generate NetFlow records from the PCAP file (in real-time) was made. This module works adding a NetFlow data stream as a new input data source to the Network Traffic Dashboard Subsystem.

10.1.3 Enabling technologies

The technologies, products, and libraries used for implementation are:

- Java[3] (1.8 version) as programming language.
- Akka library[4] (2.4.7 Version) as support library for building concurrent and distributed applications.
- Elasticsearch[5] (2.2.0 version) as NoSQL Data Base.
- Jest[6] (2.0.0 version) a Java HTTP Rest client for Elasticsearch.
- Apache Kafka (0.10.1.0 version) as message broker.

[3] https://www.java.com
[4] http://akka.io/
[5] https://www.elastic.co/products/elasticsearch
[6] https://github.com/searchbox-io/Jest/tree/master/jest

- D3.js[7] (3.5.12 version) / C3.js[8] (0.4-10 version) / Leaflet[9] (0.7.7 version) as JavaScript libraries for data visualizations.
- Spring[10] (4.2.8 version) as programming framework.
- jnetpcap[11] (1.9 version) to process the PCAP files.
- libpcap[12] (library used by jnetpcap).

10.1.4 API specification

The output interface defined for the Anomaly Detection Subsystem is a XML generator (it generates XML files periodically at specified time intervals). Each XML contains a list of attributes that define the anomalies detected in sliding time period (see section 10.1.1).

The Network Traffic Dashboard Subsystem processes the XML files as soon as they arrive. For this, two interface means are provided: through files written into a pre-defined directory or through a Web Service implemented in the Network Traffic Dashboard Subsystem.

The following Document Type Definition (DTD) describes the XML files sent by the Anomaly Detection Subsystem to the Network Traffic Dashboard Subsystem by defining the document structure with a list of legal elements and attributes. The DTD is associated to a particular XML document by means of a document type declaration (DOCTYPE):

```
<!DOCTYPE UNADA SYSTEM "/path/to/file.dtd">
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE UNADA[
<!ELEMENT UNADA (signaturesOfAnomalies*,points*)>
<!ELEMENT points (point+)>
<!ELEMENT point (val*)>
<!ELEMENT val (#PCDATA)>
<!ELEMENT signaturesOfAnomalies (signatureOfAnAnomaly*)>
<!ELEMENT signatureOfAnAnomaly (rule+)>
<!ELEMENT rule EMPTY >
<!ATTLIST UNADA start CDATA #REQUIRED>
<!ATTLIST UNADA end CDATA #REQUIRED>
<!ATTLIST UNADA file CDATA #REQUIRED>
<!ATTLIST UNADA aggreg CDATA #REQUIRED>
<!ATTLIST UNADA totalSize CDATA #REQUIRED>
<!ATTLIST UNADA totalNbPackets CDATA #REQUIRED>
<!ATTLIST point id CDATA #REQUIRED>
<!ATTLIST val dim CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly dissim CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly mainIPs CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly point CDATA #REQUIRED>
<!ATTLIST signatureOfAnAnomaly possAnom CDATA #REQUIRED>
<!ATTLIST rule dim CDATA #REQUIRED>
<!ATTLIST rule type CDATA #REQUIRED>
<!ATTLIST rule value CDATA #REQUIRED>
]>
```

[7] https://d3js.org/
[8] http://c3js.org/
[9] http://leafletjs.com/
[10] https://projects.spring.io/spring-framework/
[11] http://jnetpcap.com/
[12] https://sourceforge.net/projects/libpcap/

11. Use Case #1 Testing

11.1 Performance Evaluation

11.1.1 Relevant Metrics

11.1.1.1 Anomaly Detection Subsystem Metrics

The evaluation of the Anomaly Detection Subsystem is two-fold. It consists of evaluating both the quality of the detection (as well as the classification) of anomalies in the traffic, and the detection time (it is expected to have a fast response for being able to trigger counter measures for mitigating the anomalies).

Detection quality

The evaluation of the detection and classification quality relies on the use of classical metrics, such as TPR (True Positive Rate), FPR (False Positive Rate), FNR (False Negative Rate), and ROC curves (Receiver Oriented Curves).

- TPR is the ratio between the number of well detected (or well classified) anomalies and the total number of anomalies.
- FPR is the ratio between the number of wrongly detected anomalies and the total number of anomalies. It corresponds to a system detecting anomalies that do not actually exist in the traffic.
- FNR is the ratio between the number of undetected anomalies and the total number of anomalies. It corresponds to the number of anomalies the system was unable to detect.
- A ROC curve is the representation of the TPR depending on the number of wrong detections, with wrong detections being the sum of FPR and FNR. On such a curve, the line TPR=FPR+FNR corresponds to the performance of a random detection process. The ideal curve has the equation TPR=1 for FPR+FNR>0. The closer to this top ideal line, the better the detection system.

Note however that the detection quality parameters depend only on the algorithm and its configuration, not on implementation. For that purpose, the detection quality results are the same with the ones obtained from the experimentation of the used ORUNADA algorithm; they are presented in deliverable D4.2 [1]. For not overloading this deliverable, the presentation of the results is not duplicated here.

Detection time

The detection time is the time that elapses between the moment the first packet of an anomalous flow enters the network and the moment the detection system raises an alarm for this flow. This obviously relates to the time required for ingesting data to the system and the execution time of the detector.

11.1.1.2 Network Traffic Dashboard Subsystem Metrics

The main metrics defined for the Network Traffic Dashboard Subsystem software application are:

- The time required to export PCAP to NetFlow files.
- The time required to process NetFlow records and send them to the database.

- The time required to import all processed NetFlow packets into the database.
- The time required to execute queries to the database as a function of the size of the data stored.

11.1.2 Mechanisms

11.1.2.1 Anomaly Detection Subsystem Mechanisms

Performing quality evaluation of the Anomaly Detection Subsystem along the metrics specified in the previous section requires a set of labelled traces, i.e. traces for which all anomalies are known and labelled. This is practically a very strong constraint and really impossible to respect. Indeed, two kinds of labelled traces exist: synthetic and real.

Synthetic traces are traces that have been built for that purpose. They consist of traffic traces (real or artificially generated) in which artificial anomalies have been injected. The advantage of this approach is that all anomalies are perfectly known and classified. The main drawbacks are related to the unfortunately limited number of anomalies and anomaly kinds that can be injected and, of course, their limited realism. Examples of synthetic traces for anomaly detection include the famous KDD dataset that has been widely used for years. Its advantage is its availability, and remains today the largest dataset of this kind. On the other side, it is quite aged.

Real labelled **traces** are traces that have been collected on real commercial or public networks, and for which an anomaly detection process has been applied for detecting the anomalies contained in the trace. This process can be handmade in some cases, or rely on existing anomaly detection tools. The advantage of this kind of labelled traces is its realism, which makes it interesting for evaluation purposes. On the other side, it is not guaranteed that the applied detection process detected all anomalies and that the detected anomalies have been well classified. It can therefore lead to errors and unfair deviations when the evaluation of a new detection tool relies on such traces. Up to our knowledge, the largest publicly available dataset of this kind has been collected by the MAWI working group of the WIDE project[13] on a trans-Pacific link between Japan and USA. Traces are collected every day since year 2000 on the basis of 15 minutes of traffic collected every day, plus on some particular days, full day traces.

The Anomaly Detection Subsystem developed for UC #1 has been evaluated on these two kinds of datasets, namely KDD'99 and MAWI [1][15]. We also created our own synthetic dataset in order to include more recent anomalies and attacks than the ones included in KDD'99 [2]. This new synthetic dataset has been used for evaluating the detection accuracy of ORUNADA, and as such, the results are reported in deliverable D4.3 [2].

The project has also started spending effort to build a new labelled dataset based on the real ONTS traces captured. The process for building such a dataset is described in Section 12.1 . It consists in a collaboration with MAWIlab for first labelling the traces, and second in leveraging corrections of labels.

11.1.2.2 Network Traffic Dashboard Subsystem Mechanisms

This section analyses the performance of the dashboard subsystem by identifying potential bottlenecks requirements for system dimensioning in terms of hardware resources.

[13] http://mawi.wide.ad.jp/mawi/

As already outlined, the Network Traffic Dashboard Subsystem performs two main time-sensitive tasks: processing of incoming NetFlow data and output to Network Traffic Dashboard Subsystem.

For analysing the performance of the NetFlow storage procedure the following tasks have to be considered:

- Conversion of TCP/IP headers to NetFlow version 5 records.
- Processing NetFlow version 5 records and shipment to the database.
- Insertion in the database.

For displaying analytics information, the following considerations have to be taken into account:

- Implementation of Web Services for requesting information and returning graphical results while ensuring that the throughput is high enough to avoid information loss.
- Implementation of queries from the Network Traffic Dashboard Subsystem business logic to the database.
- Drawing procedure at the browser.

Thus, the Network Traffic Dashboard Subsystem has to support insertions in the database (from the collectors) and queries to draw analytics results (through the Web Server) at the same time. Hence, the main challenge is to make our system able to provide a fast and reliable response for these operations.

In order to test the internal Network Traffic Dashboard Subsystem performance and to detect possible bottlenecks the following tests are proposed:

- Measuring the **processing time** of the component which processes NetFlow records and sends them to the database queue.
- Measuring the **scalability** of the component which processes NetFlow records and sends them to the database queue.

The Web Server handles static documents with very high input rates. Querying a highly loaded database, however, may require longer execution times. A good performance of database queries is therefore crucial for a good user experience.

Finally, rendering the main page and charts in the browser relies on the computational power of the end-user machine. In the end, the Network Traffic Dashboard Subsystem can be regarded as a pipeline and as such all tasks across the pipeline need to execute timely enough so that to achieve near real-time performance for a good user experience.

The final requirements and the dimensioning of the architecture depend on the amount of traffic that we will analyse. A good starting point is the sizes of the ONTS files produced every day (see deliverables D2.5 [6] and D2.7 [19]).

11.1.3 Performance Tests

11.1.3.1 Anomaly Detection Subsystem Performance Tests

As stated in section 11.1.1.1 performance tests for the detection subsystem focus on detection times. Please see deliverable D4.2 [1] for evaluation tests on the quality of detection.

ORUNADA is devised to run at the border link of the core network of the Spanish medium size Internet service provider SATEC. This link has to deal with a large amount of traffic: 300,000 packets/s and 1.2 Gbit/s on average. In the context of the ONTIC project, the traffic crossing this link has been captured and anonymized since October 2014. For every packet, only the 64-bytes header is stored. The collected traffic forms the ONTS dataset and is available on demand for academic researchers. The ONTS access request form is available at the ONTIC project site (http://ict-ontic.eu/). To perform our validation, we use the file 20150210231651.pcap which contains 900 seconds of network traffic extracted on the 10th of February 2015.

Evaluation is performed on a single machine with 16 GB of RAM and an Intel Core 5-4310U CPU 2.00GHz. In the following, the window size Δt is set at 15 seconds, as ORUNADA obtained the best detection performance using this time window length (see [16]), and packets are aggregated into flows according to their IPsrc/32. We use at maximum 17 features to describe a flow. Table 2 provides the 17 features that can be obtained with the aggregation key IPsrc/32 and the 17 ones with the aggregation key IPDst/32; in total, 20 distinct flow features. The features can be modified according to the network administrator needs.

Aggregation key	Features considered	Max. number of features considered
IP source (/8,/16,/24,/32) address	- At the network level: nb of ICMP reply, nb of ICMP echo, nb of ICMP unreachable, number of ICMP time exceeded, nb of other ICMP packets, nb of distinct IP destination, total size of the flow, mean Time To Live - At the transport layer: nb of distinct port source, nb of distinct port destination, nb of URG, nb of ACK, number of FIN, nb of RST, nb of SYN, number of PUSH, subnetwork entropy of the IP destination	17
IP Destination (/8,/16,/24,/32) address	- At the network level: nb of ICMP reply, nb of ICMP echo, nb of ICMP unreachable, nb of ICMP time exceeded, nb of other ICMP packets, nb of distinct IP source, total size of a flow, mean Time To Live, subnetwork entropy of the IP source - At the transport layer: nb of distinct port source, nb of distinct port destination, number of URG, number of ACK, number of FIN, number of RST, number of SYN, number of Push	17

Aggregation key	Features considered	Max. number of features considered
IP source and IP destination address	- At the network level: nb of ICMP reply, nb of ICMP echo, nb of ICMP unreachable, nb of ICMP time exceeded, nb of other ICMP packets, total size of the flow, mean Time To Live - At the transport layer: nb of URG, number of ACK, number of FIN, nb of RST, nb of SYN, nb of Push	13

Table 19: List of the features used according to the flow aggregation key

The methodology to set ORUNADA parameters is as follows: First, we aggregate the traffic collected in a time window into flows and remove the flows which have an extreme value in at least one dimension. We use the set of flows F thus formed, to set ORUNADA parameters. To normalize the feature space, we use the max-min normalization: for each dimension, the min is set at 0 and the max at the highest value for this feature in F. As we are looking for flows whose patterns significantly differ from the others, we do not need very accurate clusters. Therefore, the length l of the distance interval in every dimension is set at 0.1, i.e. at 10% of the distance between the minimum and maximum value in every dimension in F. The minimum number of points, minClusPts, in a cluster is set at 30% of the total number of flows in F. This implies that an anomaly cannot be detected if it is made up of more than minClusPts flows. However, this is not an issue since every anomaly can be summarized in one or few flows by using an appropriate aggregation level. The minimum number of points in a dense unit minDensePts is set at 5% of the total number of flows in F.

The following evaluation aims at determining the smallest length of the micro-slot (see [1]) below which ORUNADA cannot run online. The maximum frequency of ORUNADA detection is inversely proportional to this length. The smaller the micro- slot size, the faster ORUNADA identifies the anomalies and the network administrator takes counter-measures. Thus, we have evaluated ORUNADA execution time with different micro-slot sizes. The experiments have been performed using different numbers of features and the results are displayed in Figure 16. It can be noticed that a reduction of the micro-slot size improves ORUNADA average runtime. ORUNADA can process the incoming traffic faster than it arrives as long as the micro-slot size is larger or equal to 0.3 seconds for 17 features and 0.2 seconds for 15 features. These results may be explained by the fact that few points are added, updated or removed from the feature space from one micro-slot to another (see Figure 17) as the computing is done on the global sliding window that lasts for several seconds (otherwise there would have been no gain in ORUNADA runtime with the decrease of the micro-slot size).

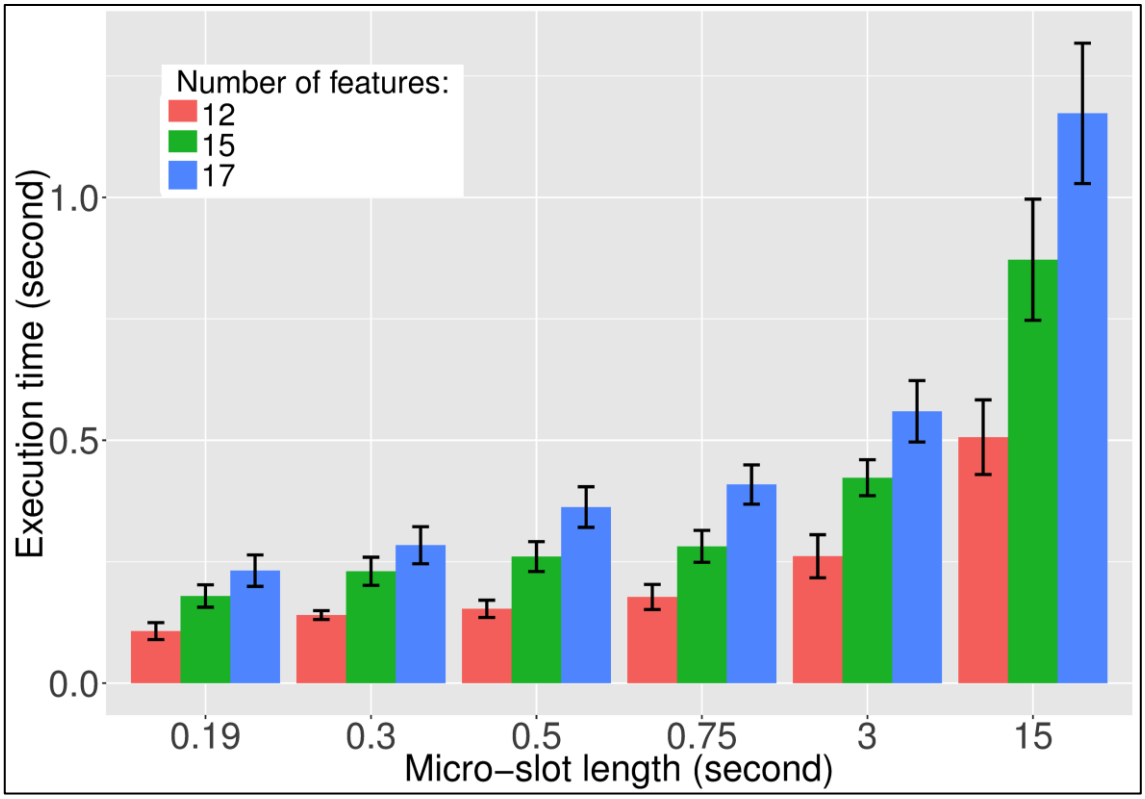


Figure 16: ORUNADA execution time according to the micro-slot length

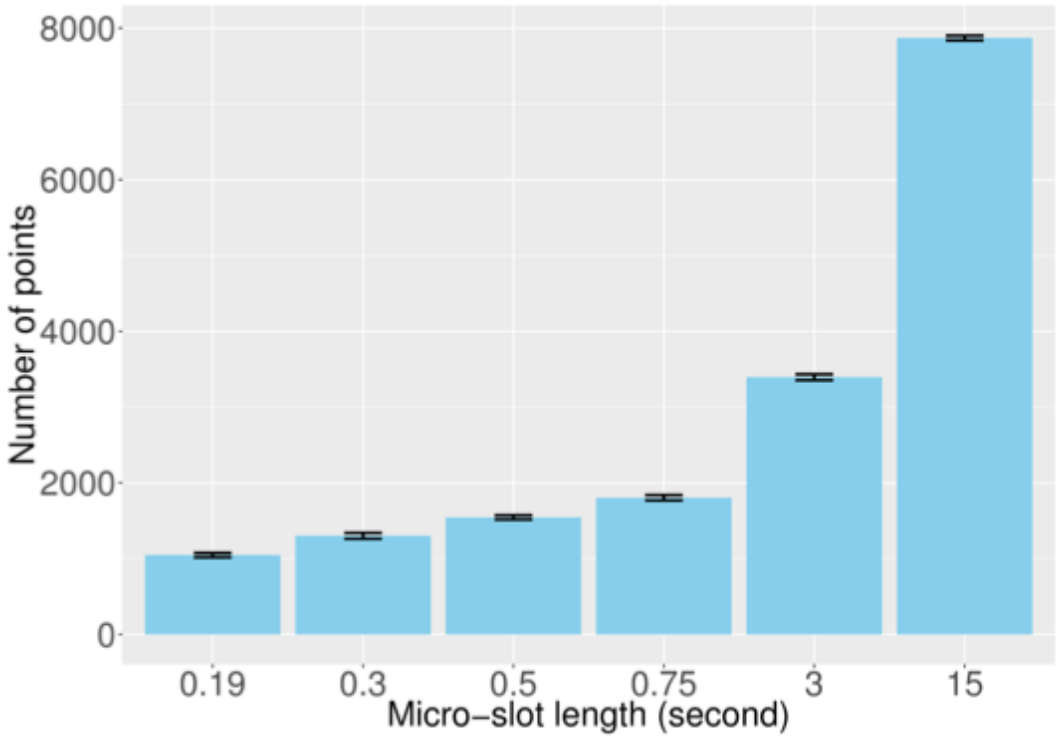


Figure 17. Mean number of points to add, remove or update at each update of the feature space partition according to the size of the micro-slot length

11.1.3.2 Dashboard Subsystem Performance Tests

As already outlined, the purpose of evaluating the Network Traffic Dashboard Subsystem performance is to assess if our architecture and related implementation technology choices can deal with high NetFlow input rates generated from a network link. The Akka-based implementation architecture for processing and storing NetFlow records (see section 10.1.2) is the key in this respect. It must prove itself to be as fast, light and responsive as necessary to achieve near real-time processing.

In order to measure Akka's promising capabilities, the Network Data Processing module was deployed in the Google Cloud Platform with the following characteristics:

- One node n1-standard-4 (4 vCPUs, 15 GB memory) dedicated exclusively to read a 10GB PCAP file, corresponding to 20 minutes of TCP/IP headers, and converting it to a NetFlow stream.
- Another node n1-standard-4 (4 vCPUs, 15 GB memory) to deploy the Network Data Processing module interface to the NetFlow exporter, i.e. the actor that listens to the stream of NetFlow records.
- Three nodes n1-standard-1 (1 vCPU, 3.75 GB de memory) where the Akka architecture can automatically deploy the remaining components of the pipeline; the parser, the filter and the writer actors.

The test consists in launching the NetFlow generator (enough resources were given so that it performs as fast as it can) and measuring the time required to process all generated NetFlow records. Figure 18 demonstrates that the Network Traffic Dashboard Subsystem architecture is able to process records at a high rate, almost at the NetFlow generation rate, which of course it is the desirable outcome.

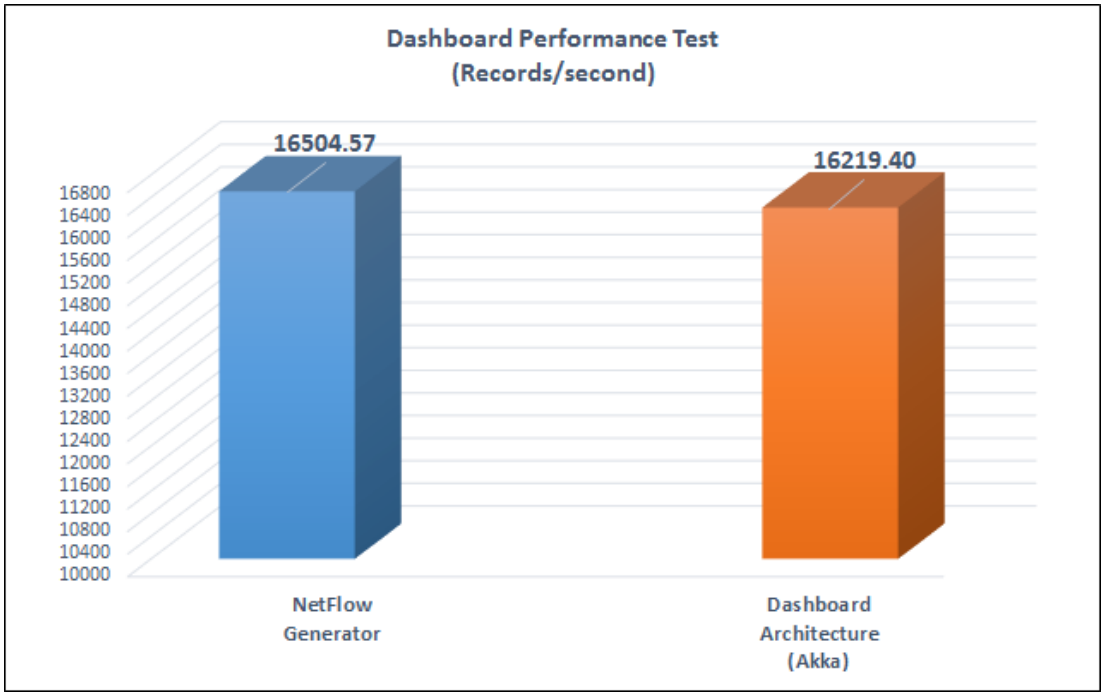


Figure 18: Dashboard Performance Test.

Figure 19 shows the automatic deployment of actors that Akka does in the available execution nodes. Keep in mind that dispositions are chosen by Akka at runtime; if we launch again the test it is possible to have a different deployment.

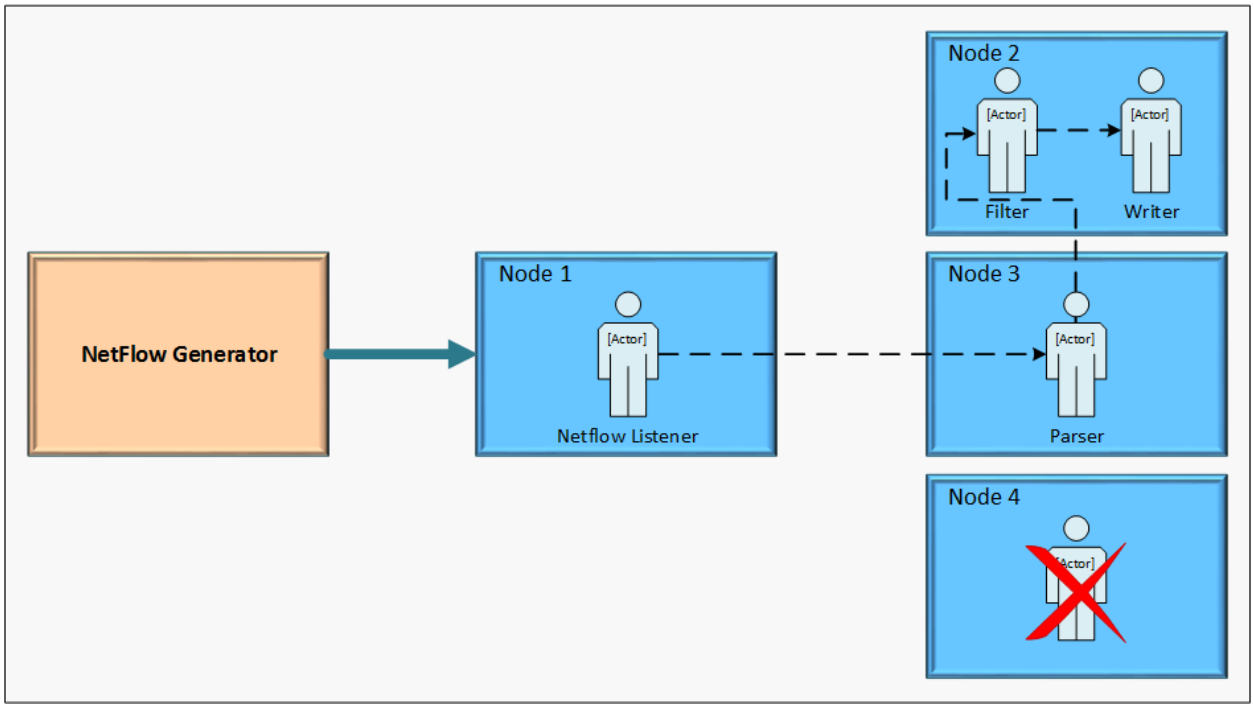


Figure 19: Dashboard performance test deploy.

Node 2, 3 and 4 have very few resources available but even so if we take a look at the actors' CPU usage, Figure 20 below, we can notice their light computational demands during the test. Obviously, node 2 has a bigger CPU usage than node 3 and the latter bigger than node 4.

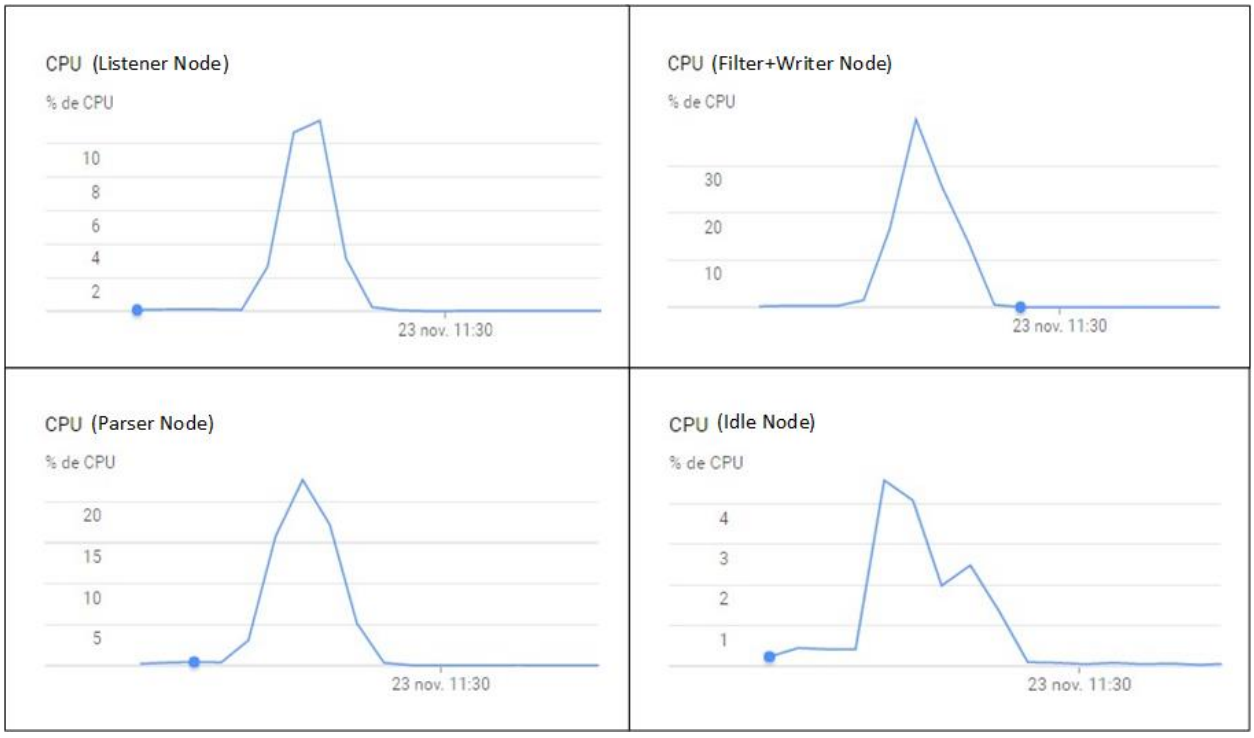


Figure 20: CPU usage in the performance test.

The exporter is able to finish processing the 20-minutes-traffic PCAP file in not much more than 2 minutes, which is pretty faster than real-time, while the architecture can efficiently sustain this incoming rate almost at the same processing times and even one of the allocated nodes was left unused (a parser actor was started in one of the nodes, and one filter actor and one writer actor in another node). Therefore it can be concluded that the architecture turns out to be extremely fast and, at the same time, keeps remarkably low resource consumption.

11.2 Scalability Evaluation

11.2.1 Anomaly Detection Subsystem Scalability Tests

ORUNADA needs to process quickly the incoming traffic to detect the anomalies efficiently. In terms of development the selection of the DGCA (Distributed Grid Clustering Algorithm) allows a significant improvement of the ORUNADA algorithm implementation.

Figure 21Figure 1 displays ORUNADA-DGCA performance. It displays the speed up factor of ORUNADA-DGCA compared to ORUNADA-DBSCAN and ORUNADA-DBSCAN with an R*-tree. One can notice that DGCA speeds up the execution time of ORUNADA by a factor of at least 300 compared to ORUNADA-R*-tree and 900 compared to ORUNADA-DBSCAN for 12, 15, and 17 features.

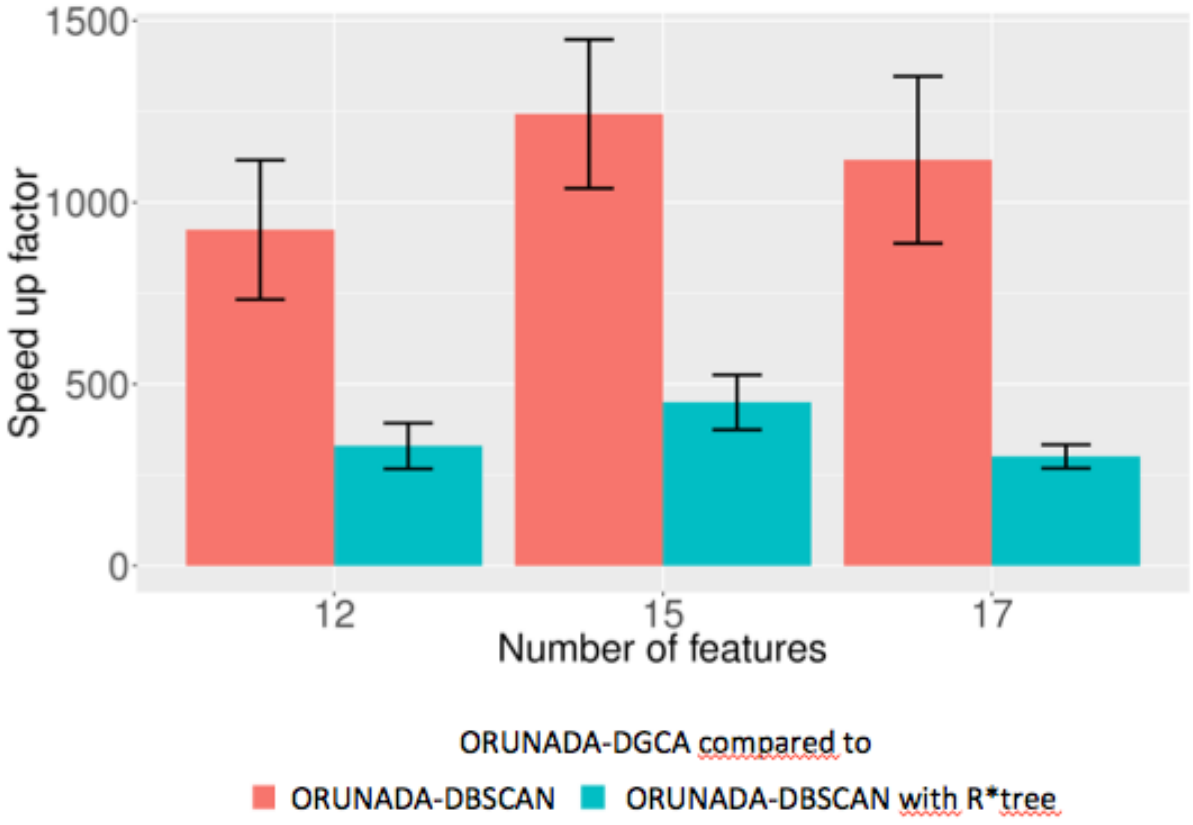


Figure 21. ORUNADA-DGCA speed-up factor

11.2.2 Dashboard Architecture Subsystem Scalability Tests

While the Network Traffic Dashboard Subsystem reveals itself to be sufficiently fast and light, it also must be ready to scale up or down in a cluster of nodes according to input load conditions. In the Network Traffic Dashboard Subsystem performance test, there is just one process providing NetFlow records but, what if there were 2, 4, 8, etc. processes generating NetFlow records? It is likely to reach a point where the architecture could not cope with higher input load; so, it would require more resources in order to keep consuming incoming data at real-time speed. Hence, the architecture's ability to scale was also put to the test.

If scalability is to be examined, a queue system is mandatory in order not to lose any record from the overwhelming incoming NetFlow input sources: Among the different alternatives, Kafka was the chosen option.

For testing scalability, the Network Traffic Dashboard Subsystem performance test is repeated but with one important difference: more than one NetFlow producers are launched and thus more than one Akka pipelines are expected to be launched too (Figure 22). Between the producers and consumers, Kafka is placed for writing/reading records to/from each part. The hardware disposition for the scalability test is as follows:

- One node n1-standard-32 (32 vCPUs, 120 GB memory) with enough resources to deploy all necessary producers.
- Other node n1-standard-32 (32 vCPUs, 120 GB memory) to deploy Kafka.
- Two nodes n1-standard-1 (1 vCPU, 3.75 GB memory) for all pipelines required.

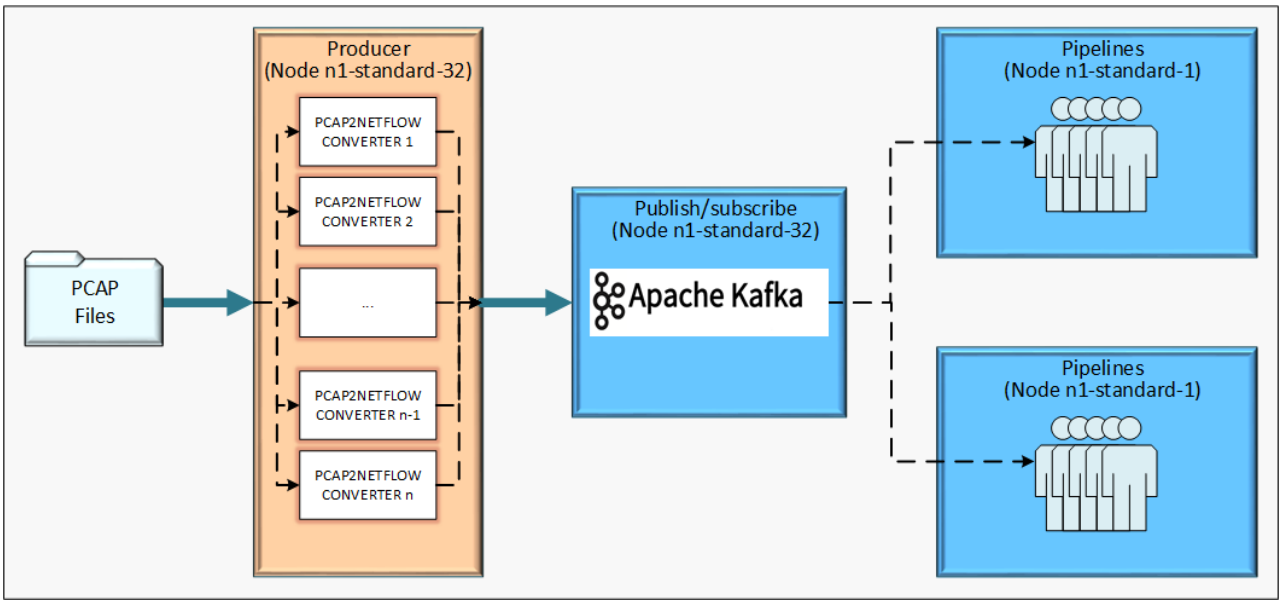


Figure 22: Scalability test architecture.

Since the machines assigned to the Akka-implemented Network Data Processing module (the last two nodes) are scarce in resources we are expecting a performance penalty, probably resulting in more pipelines initiated to match the amount of NetFlow records written into Kafka, although Akka's superb low-resource consumption could balance this out.

Figure 23 shows the behaviour of the system with a variable number of producers, from 1up to 4:

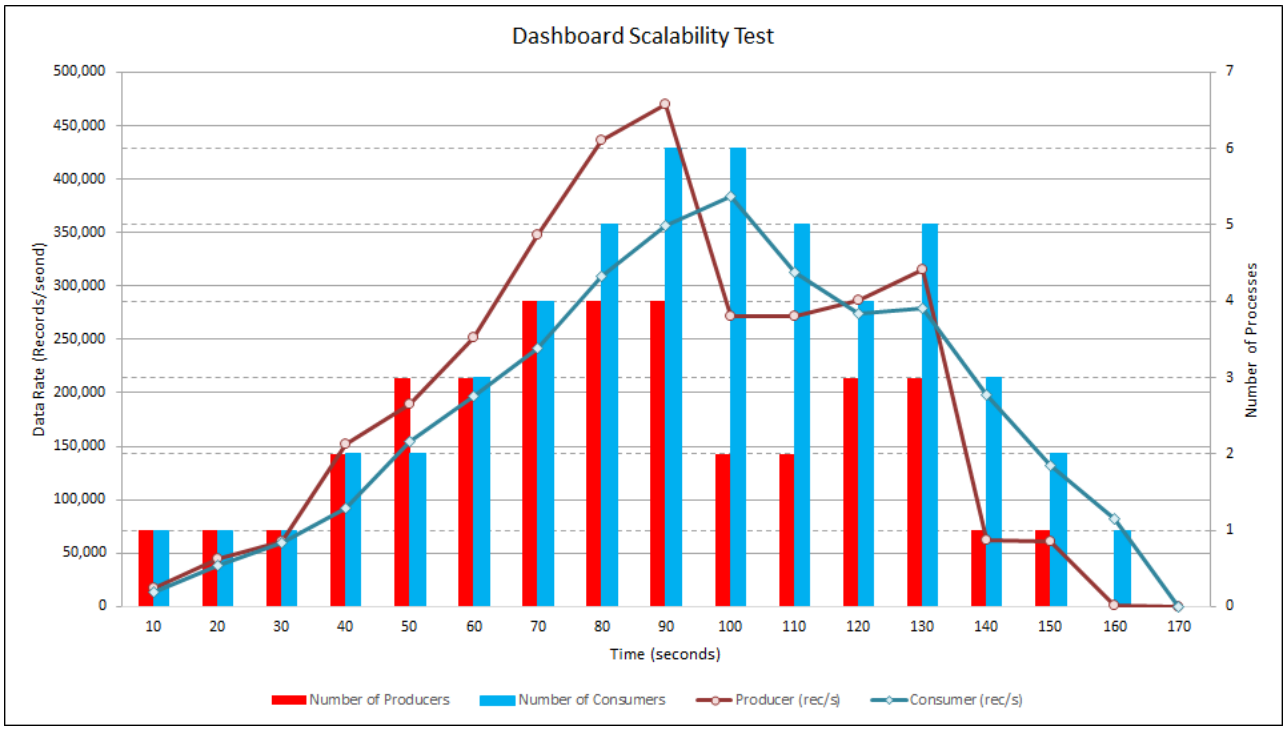


Figure 23: Dashboard Architecture Subsystem Scalability Test.

We started and stopped producer processes at will and the system responded automatically deploying or killing consumer pipelines. There is a little delay in the system's reaction time as it expands or shrinks depending on Kafka load in order to match the number of records per seconds written.

We can conclude that the Network Traffic Dashboard Subsystem shows a scalable behaviour. When input load increases, more pipelines are started so that the overall processing keeps near real-time. At the same time, when input load goes down, idle pipelines are stopped. This dynamic scalability, i.e. the ability to scale up or down on the fly all by itself without the intervention of a system administrator, is called elasticity. And, the Network Traffic Dashboard Subsystem shows such an elastic behaviour.

12. Use Case #1 Open issues, deviations and future developments

12.1 Building of a ground truth based on ONTS traces for security tools assessment

12.1.1 Lack of ground truth for network anomaly detection

Evaluation is a crucial step while building network anomaly detectors for proving their efficiency. However, it is a challenging task due to the lack of public available network data and ground truth as already outlined in section 11.1.2.1 . To our knowledge, there are two main available ground truths, the KDD99 ground truth (summary of the DARPA98 traces) and the MAWI ground truth. The KD99 dataset is quite old, and has received many criticisms mainly due to its synthetic nature [8]. But, it is still considered as a landmark in the field. On the contrary, the MAWILab data base is recent. However, its labels are questionable as they are obtained by combining the results of four unsupervised network anomaly detectors [9] and are often unintelligible, like the label “HTTP traffic”" which contains many anomalies which seem harmless after manual inspection.

In order, to overcome the lack of available dataset, researchers often build their own ground truth. We have identified three main techniques used in the literature: (a) the manual inspection of network traces [10] [11] [12], (b) the generation of synthetic traces via simulation or network emulation [13] [14] and (c) the injection of anomalies in existing network traces [10]. None of these methods are perfect. They possess their own drawbacks and they cannot guarantee accurate evaluation study; the values of true positives and negatives and false positives and negatives cannot be exactly estimated. In manual inspection neither automated algorithms nor human domain experts can identify all the anomalies in a trace with complete confidence [13]. Furthermore, due to the fuzzy definition of a network anomaly, it is hard, even for an expert, to decide when a flow becomes an anomaly, i.e. when a flow becomes rare enough to be considered as an anomaly. On the other hand, to build synthetic traces, normal traffic needs to be modelled, however, existing models often fail to catch the complexity of this traffic and the generated traffic is often not realistic. The injection of anomalies consists in injecting anomalies in existing traffic. Furthermore, the injection must be well tuned so as to obtain network traces as realistic as possible.

12.1.2 Building of two ground truths based on ONTS traces

To evaluate our algorithms, we need a recent and reliable ground truth. However, there is an important lack of ground truths in this field, which can be explained by the sensitive nature of the data. Indeed, the inspection of network traffic can reveal highly sensitive information about an organization.

In order to overcome this issue, we started to build two ground truths based on the ONTS traces:

- The first one leverages on manual inspection and injection of synthetic attacks. It corresponds to class (c) of methodologies as presented right over. The challenge is to inject a good proportion of attacks so that the data stays realistic and does not miss out or misclassify anomalies during the manual inspection. This part of the work is essential for validating the detection accuracy of anomaly or attack detection tools. The building of this synthetic ground truth is detailed in ONTIC deliverable D4.3 as it has been for the moment specifically designed for evaluating ORUNADA performance.

- The second one is based on manual inspection of some of the ONTS network traces (class (a) of the methodologies quoted above). This method is the one that proposes the most realistic traffic and anomalies/attacks, as no synthetic traffic is injected. The quality of the ground truth, on the other side, depends on the quality of the inspection and classification that has been performed. In theory, it is impossible to guarantee with such an approach that all anomalies have been classified or that the ones classified are well classified. Furthermore, this is a very time consuming task. For that purpose, we initiated collaboration with MAWILab that agrees to apply their detection and classification tools on some ONTS traces (around 4 weeks of traces) for providing a first set of labels on the traces.

12.2 Dashboard Future Developments

A full prototype of the Network Traffic Dashboard Subsystem is already available. However it would be possible to design new input data sources, new functionalities and new types of analysis in order to offer a more comprehensive tool for network administrators.

The open lines identified for further progress include:

- Analysing new input data sources supplying additional information about anomalies such as SNMP traps and logs from network appliances, anomaly signatures from data bases, etc.
- Prospecting, selecting and integrating of a real-time network traffic capturing system to be integrated as part of the product.
- Development of interfaces for integrating real-time network traffic capturing systems such as the ONTS Provisioning System developed by the project or other commercially available systems.
- Designing and implementing new analytics processes taken into account the new data sources.
- Improving fault tolerance and elasticity features taking advantage of the full range of the powerful characteristics of the Akka support library.

13. References

[1] ONTIC. “Deliverable D4.2. Algorithms Description.” Internet: http://www.ict-ontic.eu/, Feb. 2015.

[2] ONTIC. “Deliverable D4.3. Experimental Evaluation of Algorithms for Online Network Characterization”. Internet: http://www.ict-ontic.eu/, Feb 2017

[3] ONTIC. “Deliverable D5.1. Use Case Requirements.” Internet: http://www.ict-ontic.eu/, Feb. 2015.

[4] ONTIC. “Deliverable D5.2. Progress on Use Cases” Internet: http://www.ict-ontic.eu/, Feb. 2016.

[5] ONTIC. “Deliverable D2.3. Progress on ONTIC Big Data Architecture.” Internet: http://www.ict-ontic.eu/, Feb. 2015.

[6] ONTIC. “Deliverable D2.5. Progress on Provisioning Subsystem.” Internet: http://www.ict-ontic.eu/, Feb. 2015.

[7] J. Mazel. “Unsupervised network anomaly detection.” PhD thesis, INSA, Toulouse, France, December 2011.

[8] J. McHUGH, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed By Lincoln Laboratory," ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, pp. 262-294, 2000

[9] R. Fontugne, P. Borgnat, P. Abry and F. Kensuke, "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," in Proc. ACM CoNEXT, 2010

[10] A. Lakhina, M. Crovella and C. Diot, "Mining anomalies using traffic feature distributions," in Proc of the ACM SIGCOMM, 2005

[11] K. Julish, "Clustering intrusion detection alarms to support root cause analysis," ACM Trans. Inf.Syst.Secur., vol. 6, no. 4, pp. 443--471, 2003

[12] Silveira, F and Diot, C, "URCA: Pulling out anomalies by their root causes," in in Proc. INFOCOM, 2010

[13] H. Ringberh, M. Roughan and J. Rexford, "The Need for Simulation in Evaluating Anomaly Detectors," SIGCOMM Comput. Commun. Rev., vol. 38, no. 1, pp. 55-59, 2008

[14] R. Agrawel, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," Data Mining and Knowledge Discovery, vol. 11, no. 1, pp. 5-33, 2005.

[15] J. Dromard, G. Roudière, P. Owezarski, “Online and Scalable Unsupervised Network Anomaly Detection Method”, accepted for publication in IEEE transaction on System and Network Monitoring (TNSM), 2016.

[16] Jose, Lavanya, Lisa Yan, Mohammad Alizadeh, George Varghese, Nick McKeown, and Sachin Katti. "High Speed Networks Need Proactive Congestion Control." In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, p. 14. ACM, 2015.

[17] M.A. López Peña, C. Area, S. Segovia, "A "Fast Data" Architecture: Dashboard for Anomalous Traffic Analysis in Data Networks", IEEE 11th International Conference on Digital Information Systems, 2016.

[18] ONTIC. "Deliverable D2.6. Integration. Validation and Release of ONTIC Big Data Architecture and Provisioning System". Internet: http://www.ict-ontic.eu/, Feb. 2017.

[19] ONTIC. "Deliverable D2.7 ONTS Network Traffic Datastet". Internet: http://www.ict-ontic.eu/, Feb. 2017.

Annex A : Use Case #1 User stories

ID	User Stories	Definition of Done (end Y3)	Status
1.1.1	As a CSP or ISP network administrator, I want to have efficient monitoring and unsupervised clustering techniques and related analytics, so that I can autonomously classify the network traffic.	An algorithm based on sub-space clustering and recombination that copes with noise in the collected traffic, curse of dimensionality, and that can be easily parallelize to issue real-time processing	Implementation finished
1.2.1	As a CSP or ISP network administrator, I want to have mechanisms for identifying the most significant traffic attributes, so that it becomes possible to issue traffic class discrimination rules	The sub-space clustering algorithm that recombines clustering results only for significant traffic features for the detected anomalies	Implementation finished
1.3.1	As a CSP or ISP network administrator, I want to have accurate abnormality scores, so that it becomes possible to autonomously discriminate between legitimate and illegitimate traffic classes.	A function that is able to give a score indicating whether the detected anomalies are malicious or legitimate	Implementation finished
1.4.1.1.1	As a CSP or ISP network administrator, I want to get a traffic analysis visualization tool, so that I can view overall traffic statistics regarding IPs, ports, type of service, bytes, etc.	A user view (tab) that shows traffic statistics from a PCAP file (it converts the PCAP file to NetFlow, calculates statistics and shows it on the screen as a continuous task).	Implementation finished
1.4.1.1.2	As a CSP or ISP network administrator, I want to get a flow analysis tool, so that I can view precise statistics related to traffic flows, such as conversations.	A user view (tab) that shows information about flows and conversations detected into the traffic stored in a PCAP file (it converts PCAP file to NetFlow, calculates statistics and shows it on the screen as a continuous task).	Implementation finished
1.4.1.1.3	As a CSP or ISP network administrator, I want the anomaly detection tool to show a warning message whenever an anomaly has been detected, so that I can become aware of the situation any time it happens and obtain further information by accessing the tool.	A user view (tab) that shows all the traffic anomalies. The traffic anomalies are received as XML files from the function defined in User Story 1.3.1. This is a continuous process which receives, parses, and shows results as soon as each XML file arrives.	Implementation finished
1.4.1.1.4	As a CSP or ISP network administrator, I want to be able to specify the time interval the traffic analysis refers to by choosing between the last minutes (counted from current time) or a time interval specified by arbitrary start and end times and dates, so that I have a flexible way to review the traffic and get further details of any anomaly or relevant event.	Selectable user views (tabs) to choose the time interval and to show the information in the defined range.	Implementation finished
1.4.1.1.5	As a CSP or ISP network administrator, I want to be able to select a specific date and a time interval and show all the network information available (traffic and anomalies) both in dynamic mode and in static mode, so that I have a forensic analysis tool available for historic analysis in deep	Implementation of a Forensic Mode in all tabs (user views): traffic analysis, flow analysis, and anomaly analysis. This mode allows to select date and time interval and shows the information in both modes static and in continuous time through a set of buttons which simulates a video control panel	Implementation finished

ID	User Stories	Definition of Done (end Y3)	Status
1.4.1.2	As a CSP or ISP network administrator, I want to get an anomaly detection tool, so that whenever a traffic anomaly is detected I will be aware of it at once, along with its details, and I can check traffic statistics for the specific period when the anomaly happened.	A user view (tab) that shows traffic details for an anomaly in the time interval in which the anomaly has been detected.	Implementation finished
1.4.1.3	As a CSP or ISP network administrator, I want to have a set of administration procedures, so that it is possible to manage and configure different system features.	A single button view to run a demo that runs the continuous process for: reading a PCAP file, converting it to NetFlow, analysing each NetFlow register, storing information and showing results.	Implementation finished
1.4.1.4	As a CSP or ISP network administrator, I want to have a login/password authentication procedure, so that it is possible to prevent unauthorized parties from accessing the anomaly detection tool.	An entry point to the dashboard implemented as a login view. This view asks the user name and password, checks the credentials and enables or disables the use of the dashboard	Implementation finished
1.4.1.5	As a CSP or ISP network administrator, I want to have an independent user view to be shown in a shared screen (a video-wall for instance) , so that the whole team of network administrators have a general view about the network status and can realize of every incidence in real time	A view which is shown in a different window (pop-up) has been added to the application. This view shows in Real-Time a world map in which anomalies are represented graphically and in a table	Implementation finished
1.4.1.6	As a CSP or ISP network administrator, I want to be automatically informed when a new anomaly is detected by the system, so that I can reduce the time to check the anomaly and manage the alert properly.	A SNMP gauge has been developed and integrated in the dashboard architecture. This gauge is a background process which detects new anomalies and sends SNMP traps.	Implementation finished
1.4.1.7	As a CSP or ISP network administrator, I want to generate technical reports with the dashboard information, so that I can select a specific time interval and generate an electronic document including all the traffic information and anomalies in that time interval	The anomaly detail user view includes a button which generates a report in PDF format describing the anomaly selected.	Implementation finished
1.4.1.8	As a CSP or ISP network administrator, I want to export stored data in the system in a well-known interchange format, so that I can make further analysis using the dashboard data in other powerful analytic tools.	The administration console implements a functionality that enables to export all the information stored in the database about anomalies and/or traffic between two selected dates.	Implementation finished

Table 20: ONTIC Use Case #1 user stories

Annex B : Documentation of ORUNADA package

Link to the code repository: https://gitlab.com/ontic-wp4/ORUNADA

ORUNADA is a tool which detects anomalies in PCAP files in a continuous way. It relies on a sliding window and on grid subspace clustering and evidence accumulation techniques to identify in continuous anomalies and generate signatures to describe them. The algorithm is described in the deliverable 4.2 of the ONTIC project which can be found on the ONTIC site.

The clustering step is performed with IGCA (Grid Clustering Algorithm) algorithm.

B.1 Input dataset

UNADA accepts as input a folder with PCAP files. Furthermore, ORUNADA may take time (due to the extraction of the features which should be done in C to gain time) if the input files are very large and may then need a lot of RAM. UNADA looks for anomalous flows in this PCAP file.

B.2 Content of the package

This package contains:

- A file README.md: it describes the package and how to use it.
- A directory src: it contains the sources of the program.
- A file pom.xml: it provides the necessary information to MAVEN to build the project.
- A file ExampleDirectory: an example of directory with PCAP files to process.
- A jar ORUNADA.jar: a jar file to launch UNADA.

B.3 Requirements

If you want to recompile the code source, you must install MAVEN. We recommend using Apache Maven 3.3.3. To execute UNADA, with the PCAP file 'ExampleInputFile.pcap' we recommend a machine with at least 12GB of RAM to apply UNADA with DBSCAN and 8GB of RAM to apply UNADA with GCA.

You need to install the jNetPcap1.4 library

B.4 Arguments

UNADA takes 3 mandatory arguments plus 4 optional ones. The four mandatory arguments are:

- The path to the PCAP file to analyse.
- The direction of the aggregation. You have to specify whether the aggregation is made at the IP source 'src' or at the IP destination 'dst'.
- The mask of the aggregation. You need to specify if it is '8', '16', '24', '32'. The four optional arguments are:
    - Time of a slot in seconds;
    - Nb of micro-slots in a slot;

- For computing the size of the intervals for IGDCA. Each dimension has a different size of interval. This value is set as a percentage of the maximum distance between every pair of points for each dimension. To fix them, you need to specify this percentage and specify a value between 1 and 99.");
- For computing the minimum number of points to form a cluster in IGDCA. This number is set as a percentage of the whole number of points. To fix it, you need to specify this percentage and specify a value between 1 and 99."); If you don't provide the four optional arguments, some defaults ones are used.

B.5 How to run

To launch ORUNADA with the aggregation level at the IP source with the mask 32 and with no optional arguments, the command line is:

```
java -Djava.library.path=/pathToTheJnetPcapLibrary/jnetpcap -Xms7G -jar ORUNADA.jar /pa
thToThePCAPFolder/ src 32
```

To launch UNADA with DBSCAN and the aggregation level at the IP source with the mask 16 and with the two optional arguments, the command line is:

```
java -Djava.library.path=/pathToTheJnetPcapLibrary/jnetpcap -Xms11G -jar ORUNADA.jar /p
athToThePCAPFile/file.pcap src 16 15 30 5 10
```

B.6 How to compile

To compile the code source, the command line is:

```
mvn compile
```

To create a package from the code source, the command line is:

```
mvn package
```

B.7 Output

It outputs on the standard output some information about ORUNADA's execution. It also creates an XML file for each micro-slot processed (apart for the n first micro-slots, n equals to the number of micro-slots in a window). This XML lists the anomalous flows found in the PCAP file at the end of each micro-slot considering the packets contained in the current window. For each anomalous flow it specifies its features, its score of dissimilarity and its signature.

Annex C Prototype/Demo (User views)

The following sections show the different windows, tabs and views that compose the user graphical interface implemented for the Use Case #1 prototype.

C.1 Login

Figure 24 shows the login window that is the entry point to the dashboard.

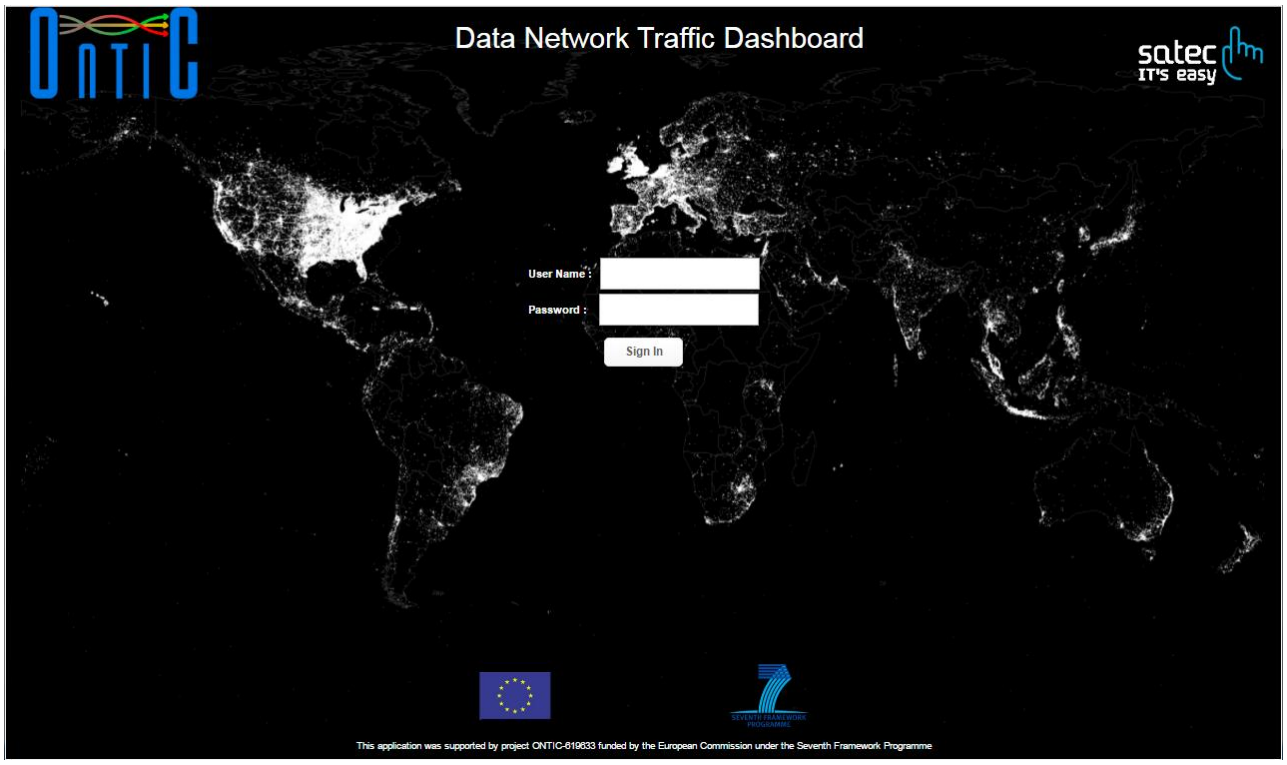


Figure 24: Login view.

C.2 Administration Console

Figure 25 shows the user interface for administration functions such as: PCAP file selection (for simulations and forensic mode), data source selection, filter level selection and data base exportation.

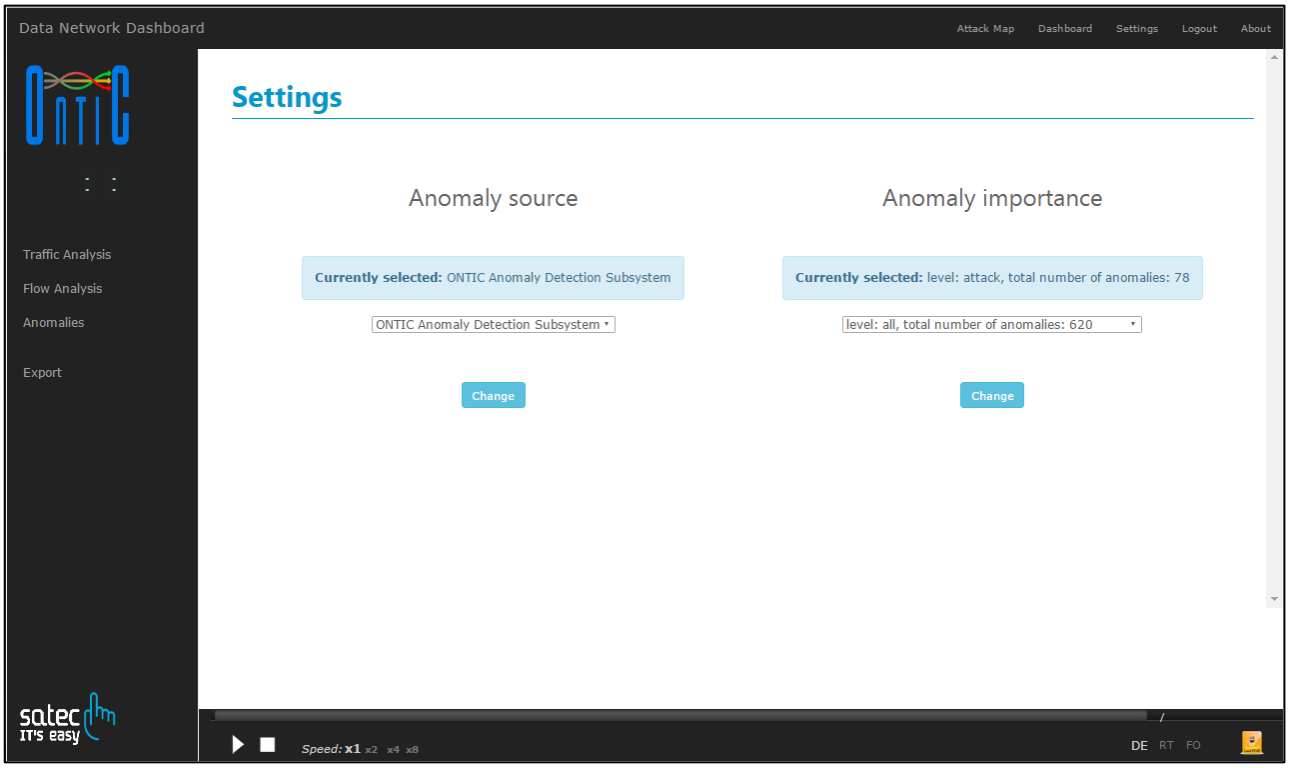


Figure 25: User view of the administration console.

C.3 Real-Time general view

Figure 26 presents a user view (which can be displayed in a video-wall and be shared by all network administrators) that shows in Real-Time a world map in which anomalies are represented in both graphical format and in a table.

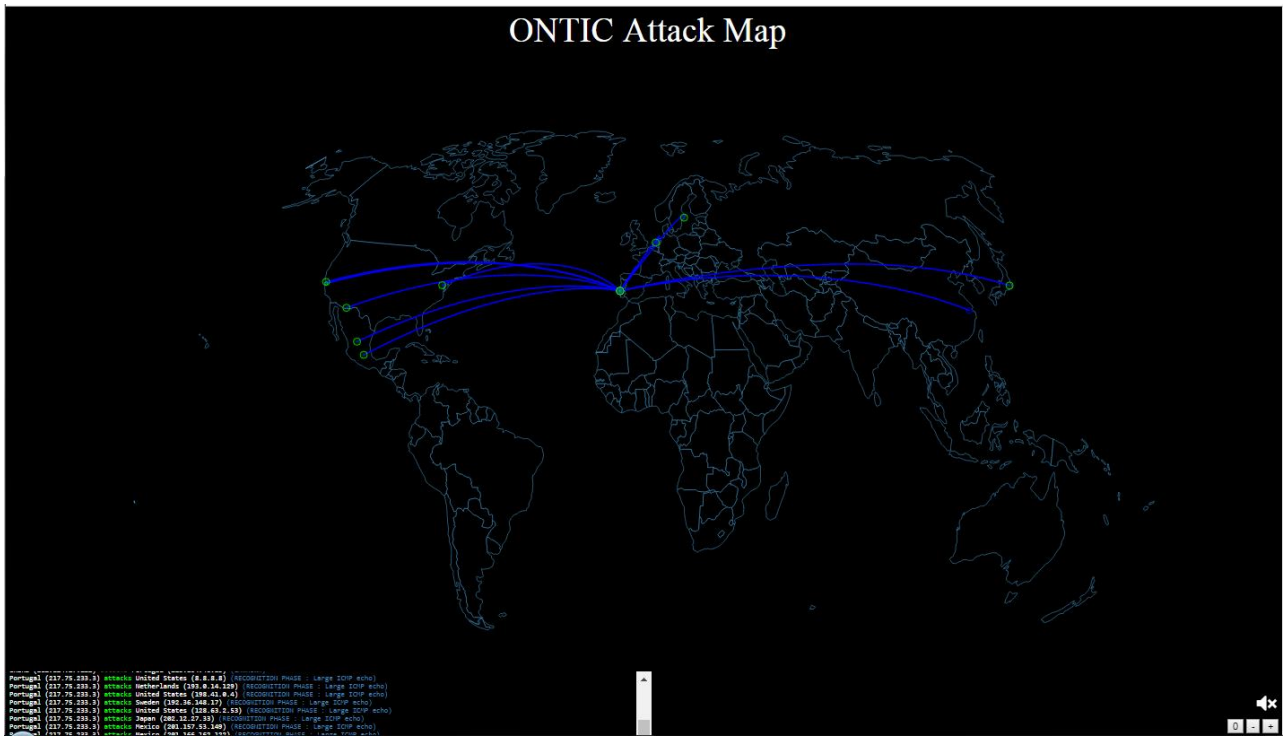


Figure 26: Shared general view

C.4 Real-Time Traffic/Flow Analysis

The following figures (Figure 27 to Figure 29) show several views of real-time traffic and flow analysis in the Network Traffic Dashboard Subsystem..

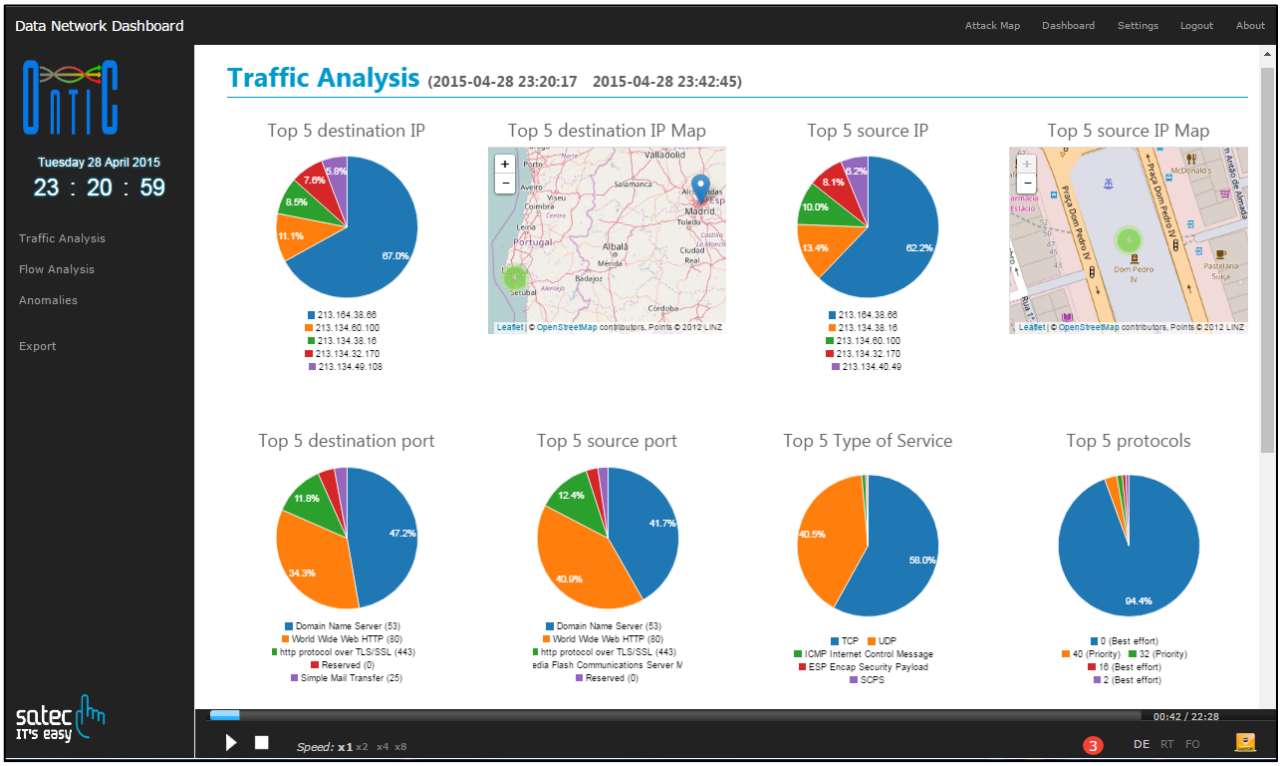


Figure 27: Real-time traffic analysis view (1).

Figure 28: Real-time traffic analysis view (2).

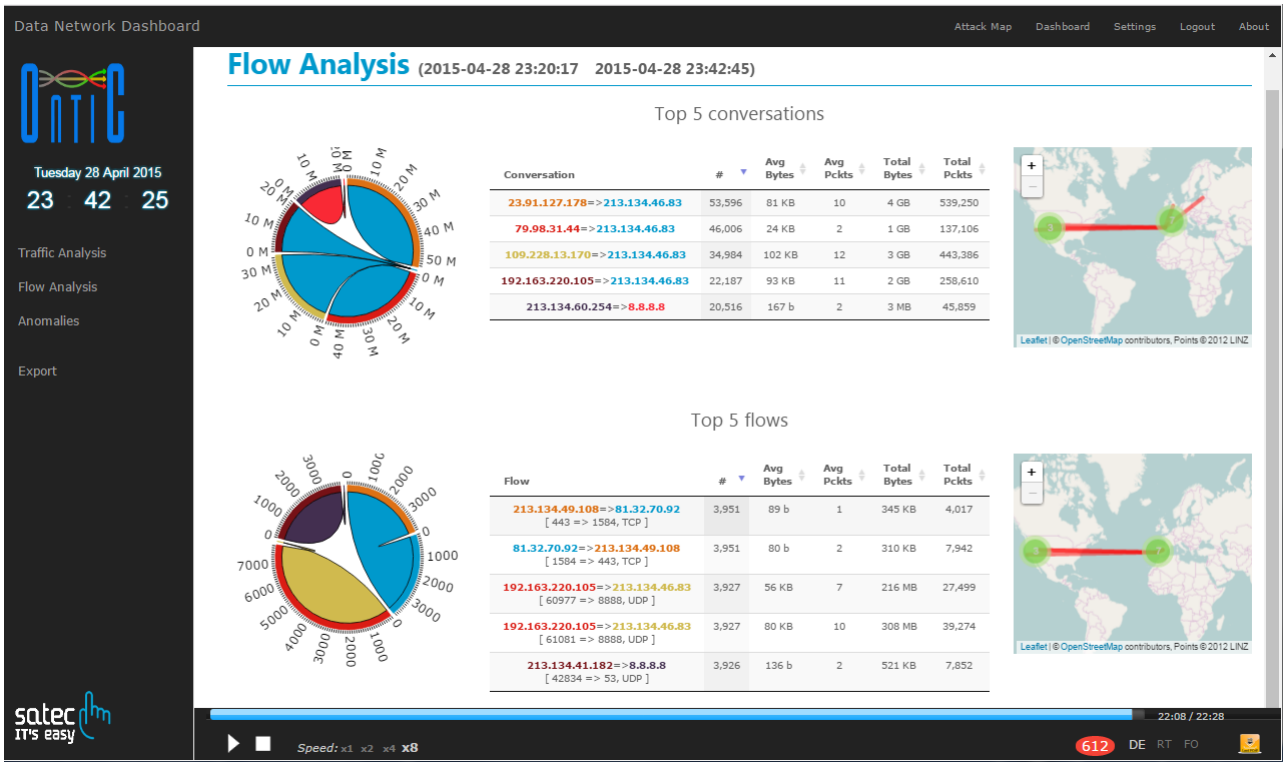


Figure 29: Real-time flow analysis view.

C.5 Forensic Traffic/Flow Analysis

Figure 30 to Figure 32 show Network Traffic Dashboard Subsystem user views for the forensic analysis functionality.

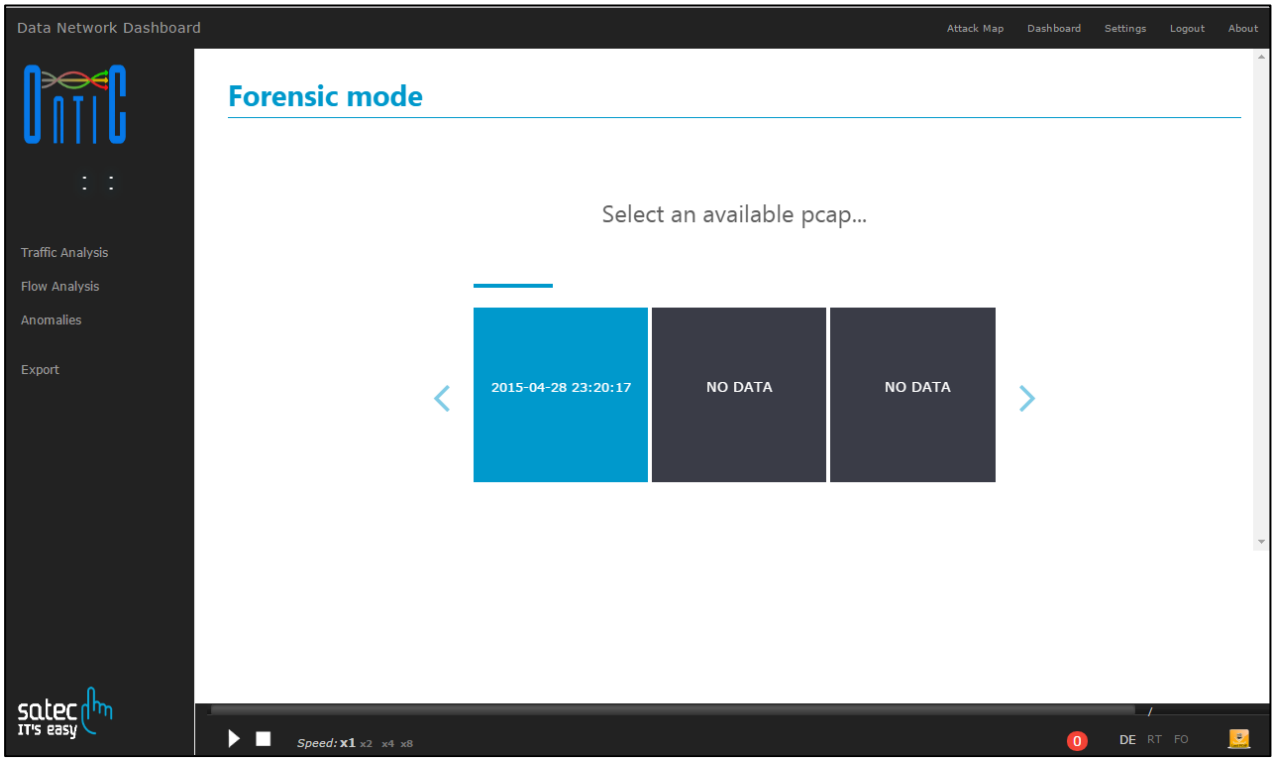


Figure 30: Forensic functionality user view (1).

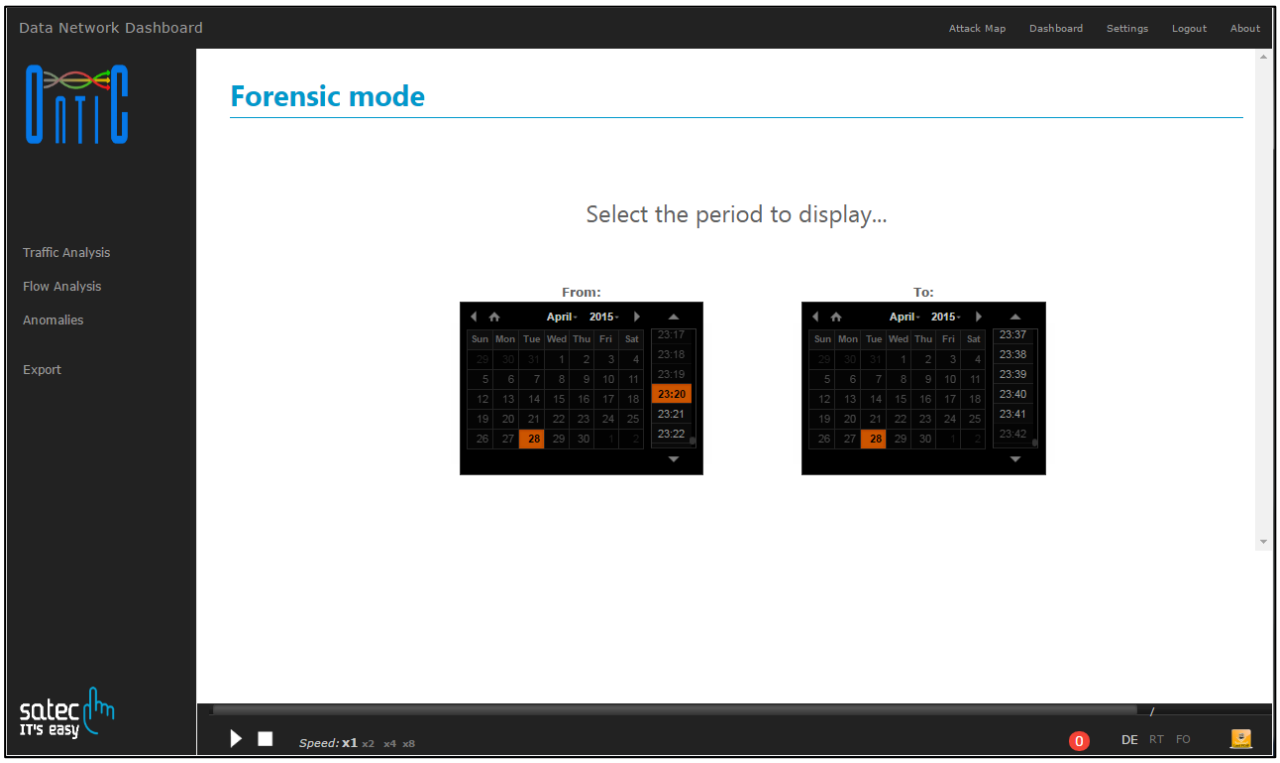


Figure 31: Forensic functionality user view (2).

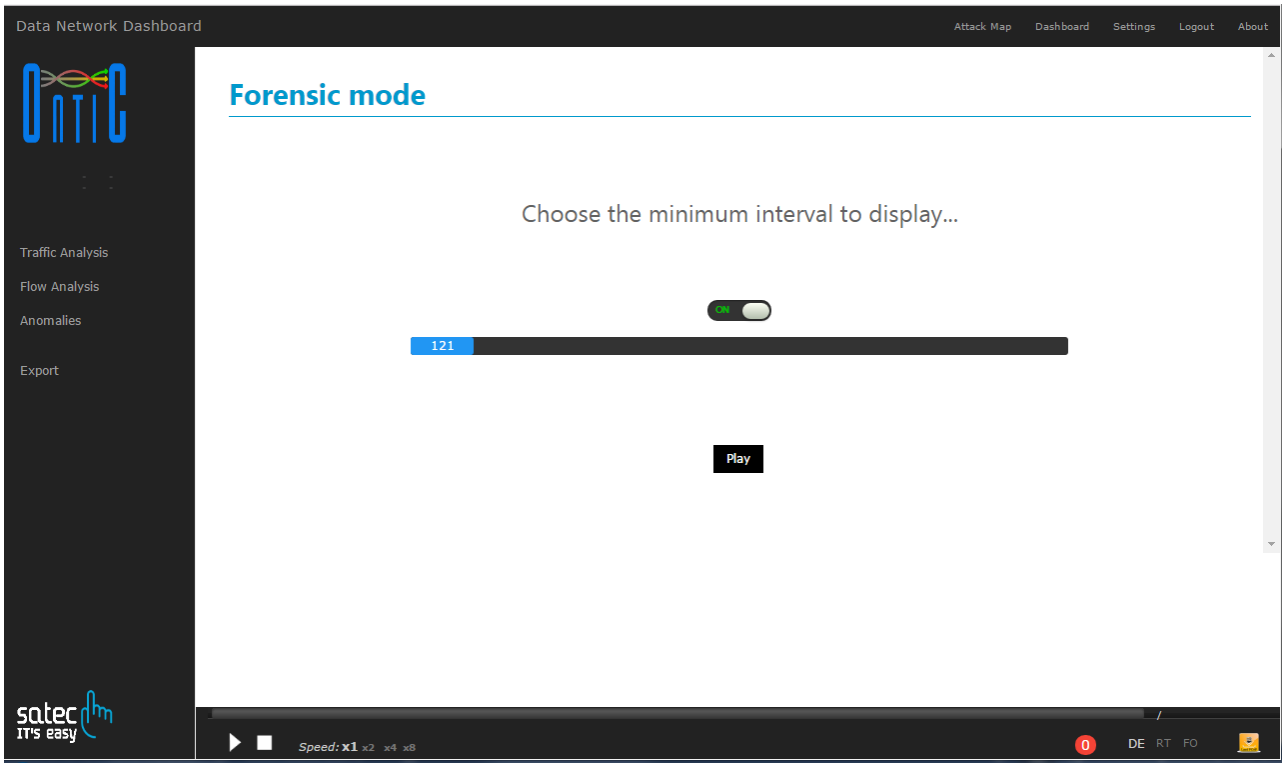


Figure 32: Forensic functionality user view (3).

C.6 Anomaly detection Analysis

Figure 33 and Figure 34 show respectively the views for the whole list of anomalies and for the detail of a specific anomaly selected.

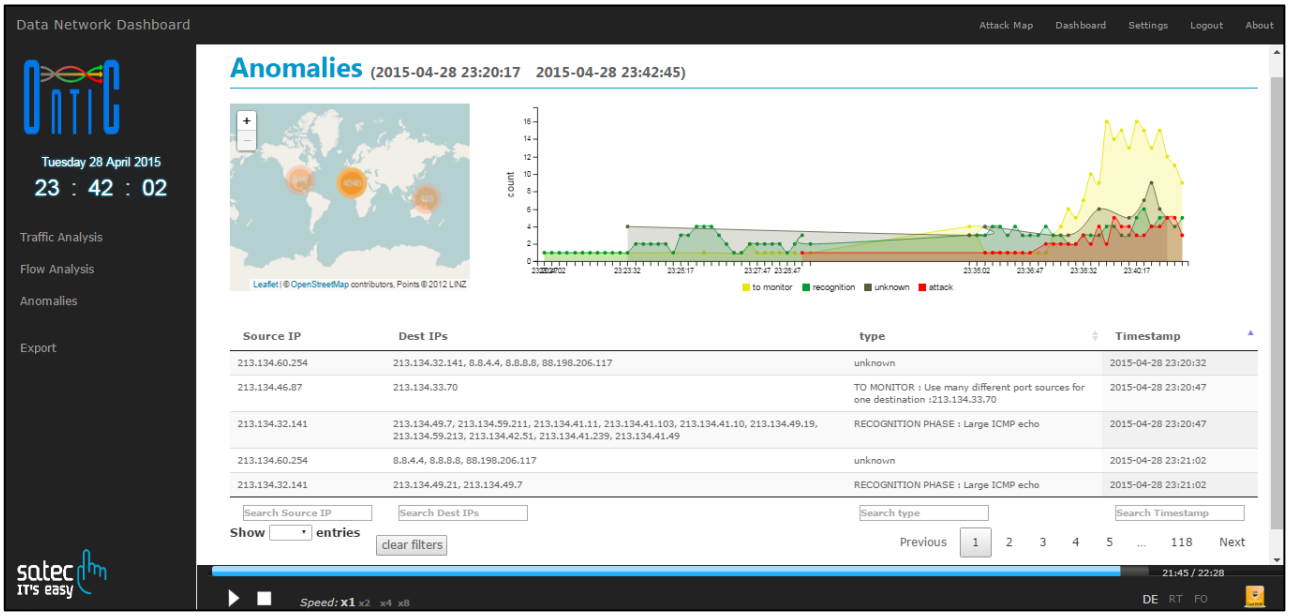
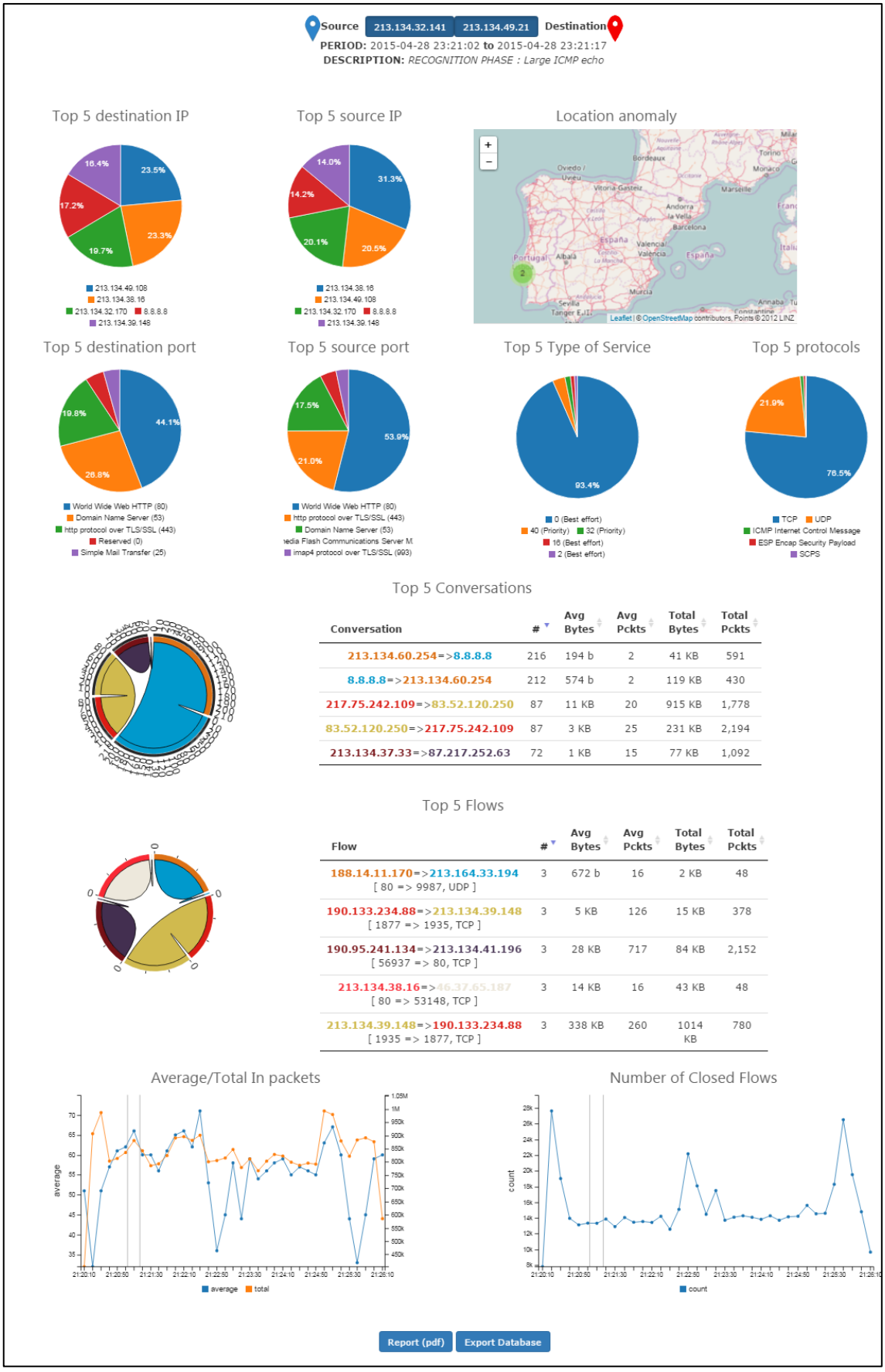


Figure 33: Anomalies user view (general).



Top 5 Conversations

Conversation	#	Avg Bytes	Avg Pckts	Total Bytes	Total Pckts
213.134.60.254=>8.8.8.8	216	194 b	2	41 KB	591
8.8.8.8=>213.134.60.254	212	574 b	2	119 KB	430
217.75.242.109=>83.52.120.250	87	11 KB	20	915 KB	1,778
83.52.120.250=>217.75.242.109	87	3 KB	25	231 KB	2,194
213.134.37.33=>87.217.252.63	72	1 KB	15	77 KB	1,092

Top 5 Flows

Flow	#	Avg Bytes	Avg Pckts	Total Bytes	Total Pckts
188.14.11.170=>213.164.33.194 [80 => 9987, UDP]	3	672 b	16	2 KB	48
190.133.234.88=>213.134.39.148 [1877 => 1935, TCP]	3	5 KB	126	15 KB	378
190.95.241.134=>213.134.41.196 [56937 => 80, TCP]	3	28 KB	717	84 KB	2,152
213.134.38.16=>46.37.65.187 [80 => 53148, TCP]	3	14 KB	16	43 KB	48
213.134.39.148=>190.133.234.88 [1935 => 1877, TCP]	3	338 KB	260	1014 KB	780

Figure 34: Anomaly detail user view.