UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Mobility of the environment and architecture of the project SECURITY at the EDGE network

Final degree project

Author:
Ferran Pérez Gutierrez

Director:
René Serral Gracià

November 26, 2018

Degree in Computer Engineering
Computer engineering

Resum

En el marc del projecte SECURED (SECURity at the network EDge),neix la necessitat d'implementar els mecanismes de mobilitat de l'entorn de l'usuari en l'arquitectura de confiança que ofereix SECURED, de forma transparent a l'usuari. La planificació ha estat acurada amb les necessitats del projecte i no s'han desviat gaire de la realitat. Els costos han estan ajustats dins de la partida pressupostaria de l'equip de la UPC. Es un projecte amb un valors de producció sostenible pel medi ambient, social i economic, segons la matriu de "la economía del bien común"[16], una proposta generada des de la FIB. Finalment la solució proposada s'ha provat en el centre d'ensenyament S'Arenal, del municipi Platja de Palma. La prova pilot ha sigut un èxit, demostrant així que la solució és exportable a qualsevol indret.

Resumen

En el marco del proyecto SECURED (SECURity at the network EDge), nace la necesidad de implementar los mecanismos de mobilidad del entorno del usuari en la aqrquitectura de confianza que ofreze SECURED, de forma transparente al usuario. La planificacion ha estado acurada con las necesidades del proyecto i se ha deviado mucho de la realidad. Los costes han sido ajustados dentro de la partida presupostaria del equipo de la UPC. Es un proyecto con unos valores de producción sostenible para el medio ambiente, social i enomico, según la matriz de "la economía del bien común"[16], una propuesta generada des de la FIB. Finalmente la solución propuesta se ha probado en el centro de enseñanza S'Arenal, del municipio Platja de Palma. La prueba ha sido un exito, demostrando de así que la solución es exportable a cualquier sitio.

Abstract

SECURED (security at the network edge) project fullfilled the need of implementing the user's mobility mechanisms in the trusted architecture that SECURED offers, in a transparent way to the user. The planning has been careful to the needs of the project and has not deviated much from reality. The costs are adjusted within the budget items of equipment at the UPC. It is a project with sustainable production values for the environment, social and economic, as the matrix of "la economía del bien común" [16] generated as a proposal from the FIB. Finally, the proposed solution has been tested in the training center of El Arenal municipality of Playa de Palma. The pilot was a success, showing that the solution is exportable anywhere.

Acknowledgments

Thanks to **Diego Montero** for his unconditional help, thanks to which I was able to move forward at the critical moments of the project. Also my thanks to **Alícia Vila** for the time she has dedicated, with a server, to put together the parts of each has developed from the project.

And finally my thanks to **René Serral Gracià** for the opportunity that he has given me, in order to be able to do the FDP, in a great project as has been SECURED and for the patience that has had with me.

Contents

List of Tables

List of Figures

Source code index

Glossary

Antiphising Anti-phishing software consists of computer programs that attempt to identify the content of identity theft contained in web pages and email or block users from being tricked. It often integrates with web browsers and email clients as a toolbar that shows the real domain name of the web page that the viewer is visiting in an attempt to prevent fraudulent websites Let's go through other legitimate websites.. 20

gunicorn Gunicorn 'Green Unicorn' is a Python WSGI HTTP server for UNIX. He is a pre-fork model worker. The Gunicorn server is widely compatible with several web frameworks, simply implemented, low resource consumption of the server, and quite fast.. 22

OpenFlow OpenFlow és la primera interfície de comunicacions estàndard definit entre les capes de control i la transmissió d'una arquitectura SDN.. 29

outlier In statistics, an atypical observation or atypical data is a value that differs so widely from the rest of the data that we think that an error has been made.. 39

plugin A connector (in English plugin, plug-in: "plug-in"), also known as extension (in English addin, add-in, addon or add-on) is a computer application that interacts with another application to contribute. Have an additional function or utility, usually very specific, such as serving as a controller in an application, to make this work a device in another program. 27

timestamps Timestamping is the use of an electronic timestamp to provide a temporary order between a set of events.. 38

VPN A virtual private network, XPV or VPN (from the initials of virtual private network) is a network technology that allows an extension of the local network over a public or non-controlled network. v, vi, 4, 7, 20, 27, 39

Chapter 1

Introduction

1.1 Description of the project

This final degree work, within modality A, is part of a larger project funded by the European Union with the participation of prestigious European universities and telecommunications companies. The project is called SECURED (SECURITY at the EDGE network) and has lasted three years of development.

One of the great contributions of the project is computer security. That's why it proposes an innovative architecture to achieve protection against Internet threats through the routing of traffic through a series of machines where each one is responsible for filtering it according to the criteria that the user has configured.

This architecture creates a trusted and virtualized execution environment that allows different actors (for example, individual users, corporate TIC administrators) to install on demand and run several applications on the device at the edge of the network, downloading them mobile devices, which guarantees the compliance of ease of specific and independent use of the device policies of security and uniform protection through different devices and personal networks[1].

This project aims to implement the mechanisms of mobility of the user environment in the trusted architecture offered by SECURED, in a transparent way to the user. The implementation of this new technology will be carried out progressively. The proposed architecture will be validated initially in a controlled test bench and finally in a real environment.

1.2 Actors involved

In project SECURED intervened a total of 7 partners. Here we can see the table with names of partners, the country of origin and the role inside of project.[2]

1.2.1 Team leader

Each partner has a team leader that manages the resources, the workload, marks the short-term goals and organizes the meetings. In René Serral Gracià is the leader of our

Initials	Full name	Country	Rol
POLITO	Politecnico di Torino	Itàlia	coordinator
HPLB	Hewlett-Packard LTD	UK	partner
PTEL	Primetel PLC	Xipre	partner
TID	Telefónica Investigación y Desarrollo SA	Spain	partner
UNICRI	United Nations Interregional Crime and justice Research Institute		partner
UPC	Universitat Politècnica de Catalunya	Spain	partner
VTT	VTT Technical Research Centre of Finland	Finland	partner

Table 1.1: Consortium

team and the director of my final degree project.

1.2.2 UPC team

The role of the UPC in the project is as follows:

- Definition of the requirements for architecture, as well as its design and evaluation.
- API definition, ontologies and summaries policies, both the security requests and the configurations required at different levels, with the participation of both physical and virtual devices.
- Orchestration and development to attend mobile users.
- The creation of prototypes and integration.

The roles that directly affect me are those of orchestration and creation of prototypes (migration) and integration.

1.2.3 Other partners

I will now briefly explain the role that other SECURED project partners play:

1. POLITO
    - Coordinator of the project SECURED.
    - Development and research in security policies.
    - Development and research in software and applications for the edge device of the network, based on its experience in the speed of flexible processing and high network traffic, formal methods of verification of security code, and the recent work of the network, Explore the migration of personal network applications from the host to the edge of the network.
2. HPLB
    - Carry out the general definition of the technical architecture to ensure that it will be commercially relevant and realistic.

- Conduct the development and implementation of the fundamental edge architecture network device.

3. PTEL

   - The development and research activities in the design and evaluation of the architecture of SECURED, contributing to the requirements of the network from the perspective of an operator.
   - Collection of information and feedback from end users.
   - Technical validation of SECURED.
   - The dissemination of project results.

4. TID

   - Contribution to the definition of the needs and options of architecture for scenarios of network operator.
   - Contributing to the design of the security service, in front of heterogeneous scenarios, mixes legacy network equipment and new edge nodes, which also incorporate the operator's safety regulations and internal regulations.
   - Provide test environments with the actual HW/SW in use within the Telefónica, and simulate different types of traffic mobility.
   - They actively participate in standardization through the TID bodies. They do the analysis, evaluation and validation of the feasibility of proposals in the broadest Internet environment.

5. UNICRI

   - Assisting in the payment requirements of the interested parties, paying special attention to three main areas of risk: in line with the protection of children, the fight against computer crime, the continuous support of Internet users against authoritarian regimes.
   - The execution of the evaluation process with the interested parties.
   - Contributing to the specification of the policies on the balance of the fight against computer crime and privacy, the protection of children online and support the network against censorship and vigilance.
   - Evaluation of the ability to use the user interface for the specification of the policy.

6. VTT

   - The investigation in the areas of detection of intruders and anonymously.
   - That brings all the integration work.
   - Organize a lab for integration.
   - Diffusion and exploitation through national and international VTT networks.

1.2.4 Beneficiary

The direct beneficiaries of the projects are the telecommunications companies themselves that are partners of the project. Indirectly, the universities involved may use part of the project in new research channels. In addition to the project once it is finished it will be made public and will have access to it.

1.3 State of the art

This project aims to create a product that offers a guarantee of security and portability in any environment. Therefore, much of the architecture that arises is to use virtualization resources and routing policies based on the Open-FLow[3] standard.

1.3.1 Open-Flow

Open-Flow is an open standard that allows researchers to execute the experimental protocols on the campus networks we use every day. Open-Flow is added as a feature of commercial Ethernet switches, routers and wireless access points, and provides a standard to allow researchers to perform experiments, without the need of vendors to expose the internal functioning of their network devices. Open-Flow is currently being implemented by major suppliers.

1.3.2 Virtualization

Most user management and environment control machines are virtual machines using kvm, a solution based on the Virtual Machine core or KVM. It is a solution to implement complete virtualization with Linux, consisting of a kernel module (named kvm.ko) and tools in the user space, being completely free software. The KVM Component for the kernel is included in Linux from version 2.6.20.[**?**] KVM allows you to execute virtual machines using images of hard drives that contain Operating Systems to Modify. Each virtual machine has its own virtualized hardware: hard drives, graphics cards, etc.

Additionally we will use Qemu, it is a software processor terminal emulator, which allows a user to simulate a complete system within another. It's free software and it's written by Fabrice Bellard. QEMU is a hypervisor and is similar to other projects such as Bochs, VMware Works and PearPc.[4]

1.3.3 VPN

Connectivity between the user and the environment of SECURED will be performed by a tunnel (VPN). Therefore, StrongSwan will be used. It is a complete implementation of Internet Protocol Security (IPsec) for Linux 2.6, 3.x and 4.x higher. The objective of the project is to have strong authentication mechanisms through public key certificates X.509 and optional secure key storage on smart cards through a standardized PKCS[5] interface.

The extension MOBIKE[18] IKEv2 (RFC 4555) will be used, which allows to change the end of the tunnel generated in the network. And the mysql plugin to support multiple connected users.

In the chapter 6, you will learn more about how it has been implemented.

1.3.4 Opendaylight

OpenDaylight[7], is a current open source SDN (Software-Defined Networking) platform. It will be a temporary solution used initially in the project to orchestrate a migration of a virtual machine and to prepare the test bench.

1.4 Document structuring

Below is a brief summary of how the document is structured.

Scope: Chapter where the problem is formulated. Later, the objectives are explained to solve it, the scope of the project, risks and possible solutions that may arise during the course of the project and the method of work to be used.

Planning: Chapter where the tasks carried out in the development of the project and the period devoted are defined.

Architecture: Chapter where a summary of the architecture of the project SECURED will be made, functionalities prior to implementation and what was the scenario before initiating the implementation of the mobility.

Implementation: Chapter where technical depths will be discussed in all the implementations carried out in the project.

Test suite: Chapter where all the test suite will be presented, what data will be collected, under what conditions and the characteristics of the test environment where they will be made.

Results: Chapter where I will show the graphs obtained from the test games explained in the previous section.

Budget: Identification and estimation of costs.

Sustainability: Chapter that will explain in detail the environmental, social and economic advantages that the project would bring, once finalized, applied in a real situation.

Conclusions: Final chapter where it will be shown to where the project has arrived and future developments.

Chapter 2

Reach

Next, the requirements that the project must fulfill to justify the investment of time, economic investment and effort to achieve the objectives successfully will be explained.

2.1 Formulation of the problem

Protection of mobile devices from Internet threats is usually achieved through the installation of the appropriate tools (for example, antivirus, personal firewall, parental control) on each device.

This, however, raises several problems, usually requires privileged access to the device, and many times the appropriate protection tools are not available for all platforms.

Another problem observed is the software that offers a security service. Normally it is specified in each device. With a computer desktop computer, including laptops, the software can usually cover the needs, as long as the connectivity is in the same network. Therefore, just that we change the network, this application can not always guarantee security.
On mobile devices, we note that software is usually limited. In addition, these tools can consume many resources, especially when connected to mobile networks, currently the cost of data consumption is high. Mobile devices are constantly changing from one network to another, regardless of the company's wireless access point, such as the network of the telephone operator that has been hired. The user does not know and does not know that the network change is vulnerable to their device.

An emerging problem is to protect younger generations from threats or access to content that is not appropriate for their age. Legal guardians do not have any tools to control and protect this sector from vulnerable society.

All this translates into insufficient protection for users with high mobility, where in their devices it can not be guaranteed secured or controlled connectivity, if the user is an underaged.

2.2 Goal

The project SECURED proposes an innovative architecture to achieve protection against Internet threats by downloading and implementing security applications on a programmable device near the user's device, such as an access point or a router of the company.

The SECURED architecture creates a trusted and virtualized environment that allows the execution of different actors (for example, individual users, corporate IT administrators, network providers) to install on demand and run multiple security applications on the device that will make the point of access to the network, to protect the traffic of a specific user. This solution reduces the load and consumption in mobile devices, which guarantees the compliance of the specific and independent user policies of the security device, and uniform protection through different devices and networks.

Mechanisms of transition between networks are also defined to support traditional network devices and implement this new technology incrementally. It must be ensured that any architecture of the environment that sustains the user's session can be replicated when the user connects to another access point to the network. This implies migrating the entire virtualized environment at moment. All the routing policies of the user's data must be regenerated in the destination where the entire user environment is migrated.

The project will also be able to enforce the security policies on demand, not only when the employee is connected to the company's network, but also when it is in motion.

2.3 Requirements

2.3.1 Technologies

The connectivity between the user and the environment that offers SECURED must be encrypted, for this reason a secure connection will be made VPN.
Kvm and Qemu virtualization technologies must allow rapid migration of the environment.
The routing rules are easy to replicate from one access point to another.
The user must be transparent and do not have to notice the changes when changing the access point.

2.4 Reach

The result of the project is to offer a comfortable, transparent and secure solution to connect to the Internet. The user shouldn't be responsible for security, the environment that would be offered by a telephone operator. And it will have total mobility without worrying about losing connectivity, with the added security offered by the environment.

2.5 Risks and possible solutions

During the project, some problem situations may occur. For each of these an attempt will be made to provide an appropriate solution:

2.5.1 Increase in debugging time

It is a research project, therefore it is made of all the newest tools possible to develop the project, this entails a fairly high time in debugging the correct functioning of the whole ecosystem.
Solution: Set the worst scenario and always work with the latest stable versions of the technologies that we will use and give huge margin at the time of debugging before consuming the time before the next step.

2.5.2 Bureaucracy

The project depends on the financing of the European Union, each partner has a different budget. If UPC equipment is needed to obtain new hardware that surpasses the budget, then we would be talking about a limitation at the same time to be able to carry out tests.

Solution: Adjust yourself to the maximum with the possibilities that give the budget item for each partner. Justify obtaining new material to make an estimate of whether this material will be useful until the end of the project. If necessary, recycle the equipment, which initially had a function that they no longer need to develop, to allocate them to other functions that yes it is necessary to prioritize to optimize resources.

2.5.3 Test machines

The whole environment must be tested in a series of hardware that must be in perfect condition, because if it fails, a problem arises when it comes to developing.

Solution: First prove that the hardware is in perfect condition to build a test environment. With a clean operating system, without any configuration, except what you need to perform the tests.

2.6 Identification of laws and regulations

Regarding the laws and regulations, this project does not add to the current system, since the project is based on building a system that guarantees the security of the user's data. Therefore, the laws of the LOPD [6] type are respected because the sensitive data of the users are never revealed and once the user is disconnected they are destroyed.

2.7 Work method

2.7.1 Semi-presential development

Most of the project will be carried out in the office, since the machines are in a test environment that is only accessible from the UPC facilities, so that the integration tests have been done in person.The development development has taken place at a distance, with the

disadvantage that the verification of the new implementations have been done in person on the test machines.

2.7.2 Follow-up meetings of the UPC team

Regular follow-up meetings are held, at the UPC internal team, with the team leader in order to check the evolution of our part within the framework of the project SECURED. These meetings will present both advances and possible problems that may arise and actions to solve them will be set out to achieve the objectives.

2.7.3 Follow-up meetings of the SECURED project

Monthly monitoring meetings are held, where the project leaders of each team will meet with the director of the project to explain the work done in each part of the project. The progress of what is still under development. Finally, the following objectives are detailed with the next meeting.

2.7.4 Testing and integration of the environment

1. The test environment that will be mounted to integrate the mobility of the architecture SECURED, will be initially an independent environment. Therefore it will be carried out in a controlled and limited environment.

2. Once the system is consistent, it will be integrated with the rest of the components of the project, such as the PSAR[1] or the communication with the client. In the architecture chapter of SECURED, we will learn more in detail.

3. Once the other components work on the process of debugging the entire architecture to optimize the migration of virtual machines, the environment and the reconnection of the user into the environment in a new access point.

4. Collect data to plot a graph of time in the migration.

5. Finally, there is a final demonstration with the entire ecosystem, in a real environment. It can be an institute, university, etc.

Chapter 3

Planning

3.1 Temporary planning

In this section we will briefly explain the tasks that have been carried out during the development of the project. The tasks are described in chronological order and fully deepened in the implementation chapter. Finally we can graphically see the tasks with the gantt diagram, showing the days that have been assigned to each task.

3.1.1 Description of tasks

Period of adaptation to the environment of the project SECURED

The first weeks are basically a study of the architecture of the project SECURED[1], to know in what state the project is and see what the next tasks will be, such as the use of virtual machines to work with the environment. Also what are the programming languages that will be used. In this case it will be mainly Python to develop the entire process that involves the migration of the user environment and the configuration of the data routing policies.

Preparation of the testing environment

Once the knowledge of the tools and the environment has been obtained, the hardware and software must be prepared to carry out the development and subsequently the mobility tests of the SECURED project. Two NUC[17] format computers and a portable computer will be used. The NUCs are configured as an access point. These devices are connected to each other, to simulate a local network, only visible between the computers. These teams will act as NEDs.
To test the operation of the testbed, a virtual machine is used as an orchestrator to simulate the migration of a virtual machine, with opendaylight.[7] The openvswitch is used to perform the data routing test.[8]

Strongswan configuration

This task is to install and configure Strongswan. Version 5.4.0 [5] has been used. The Mobike module has been used, so it has been reconfigured to listen to Port 4500, instead

of Port 500. This task is explained in more detail in the chapter on implementation.

Integration of mobility to the project SECURED:

The most important task, which has covered most of the project, has been the implementation of the new NED class for the mobility of the user environment (TVD), with its PSAs and the PSC.
Finally, in the destination NED, openflow routing policies are regenerated, thanks to information transmitted from the NED origin. And the virtual machines in destination. Much of this block of project tasks is explained in more detail in the chapter on implementation, however, a brief summary will be made below.

First version of the migration script for virtual machines: First of all, a simpler version will be made to test the operation with a simple script or class with Python. It will be tested in the testing environment. Once it works, it will be passed to the next task.

Tests with different types of migrations: Proofs will be made with the previously created script making calculations of the time it takes, making incremental copies of the hard disk from source to destination, or creating a new hard drive. Temporary or persistent migration to destination. Convert virtual machines into "readonly", etc. All this will be used to see which is the most optimal option to carry out in the mobility of the SECURED project.

Adaptation of the VM migration code: Once the option with the least cost of time and resources is found, a new class will be created by the orchestra of the NED, called TVDMigration, which will be responsible for migrating when it is time.

Schedule the migration of the TVD instance: Once migration of the virtual machines between the NEDs has been achieved, the user's TVD must be migrated and reinstalled. To perform tests in the test environment, a Python script is improvised to simulate the user application, to simulate communication with the NED to perform the migration.

Integration of mobility with the user application: The user application sends a message REST[9] json[12]. Once received, the new TVDMigration is run, which is responsible for migrating the entire environment (PSAs, PSC, variables of the user of the TVD user). First, PSAs begin to migrate, once migrated, the PSC is migrated and finally a REST json message will be sent with the user's TVD.

Once all previous events end, the response is sent, and then the application drops the strongswan tunnel to the source, then reconnected to the new access point and finally generates a new connection to the destination.
This point will be explained in detail in the chapter 6.

Improvement and paralelization of the mobility code: Once the mobility is integrated with the application, the communication between the two parties improves, and the process of migration of PSAs is paralleled and a stable flow of the mobility process is created, which will be the following. A migration thread starts up, within this primary thread, secondary threads are started, one per PSAs, which simultaneously perform the migration of PSAs. Once these secondary threads reach 80% of the migration, they send an event to the main thread, this simultaneously initiates the secondary thread to realize the migration of the PSC. At this point, the original NED sends to the destination NED, through a message REST json the user's TVD. Once they finish and perform the join, all the threads with an event continue the execution to send the answer to the client's application.

Optimization of the communication between the application and the NED: Once the goal of optimizing migration execution time has been achieved, it is necessary to improve the communication of the application and the NED because in the case of multiple users to work perfectly. This task is explained in more detail in the implementation chapter.

Debug the code for the demonstration of Mallorca: In September 2016 a real test is done in Mallorca, therefore all the resources are focused on maintaining the entire mobility environment stable.
In the annexes there is a link to the news about the pilot test carried out in the S'Arenal, on the website of Diario de Mallorca

Data collection to chart a time graph in migration

The code is modified to add timestamps to obtain, in the most accurate way, the times in the migration of the environment. This point is deepened in the chapter7 and in the results section.

Debug the code for publication

Finally, the code of comments is cleared, unnecessary lines of code, the writing of the technical documentation that is requested to be delivered to the members of POLITO is made to put it in the public domain.

3.1.2 Gantt diagram

Project	Expected start date	Worked Days	Expected final date
1. period of adaptation to the environment	01/06/2015	29	30/06/2015
2. Test-bed spraying	01/07/2015	30	31/07/2015
3. Strongswan configuration	01/09/2015	29	30/09/2015
4. Integration of mobility to SECURED	01/10/2015	365	30/09/2016
4.1. First version of the migration script	01/10/2015	45	15/11/2015
4.2. Tests with different types of migrations	16/11/2015	45	31/12/2015
4.3. Adaptation of the VM migration code	01/01/2016	62	03/03/2016
4.4. Schedule the migration of the TVD instance	01/03/2016	31	01/04/2016
4.5. Integration of mobility with the user application	04/04/2016	29	03/05/2016
4.6. Improvement and paralelization of the mobility code	01/05/2016	45	15/06/2016
4.7. Optimization of communication	15/06/2016	46	31/07/2016
4.8. Debug the code for the demonstration of Mallorca	01/09/2016	29	30/09/2016
5. Code improvement to support multiple users	03/10/2016	12	15/10/2016
6. Data collection	17/10/2016	29	15/11/2016
7.Debug the code for publication	15/11/2016	46	31/12/2016

Figure 3.1: Table of tasks, start date, duration and end date

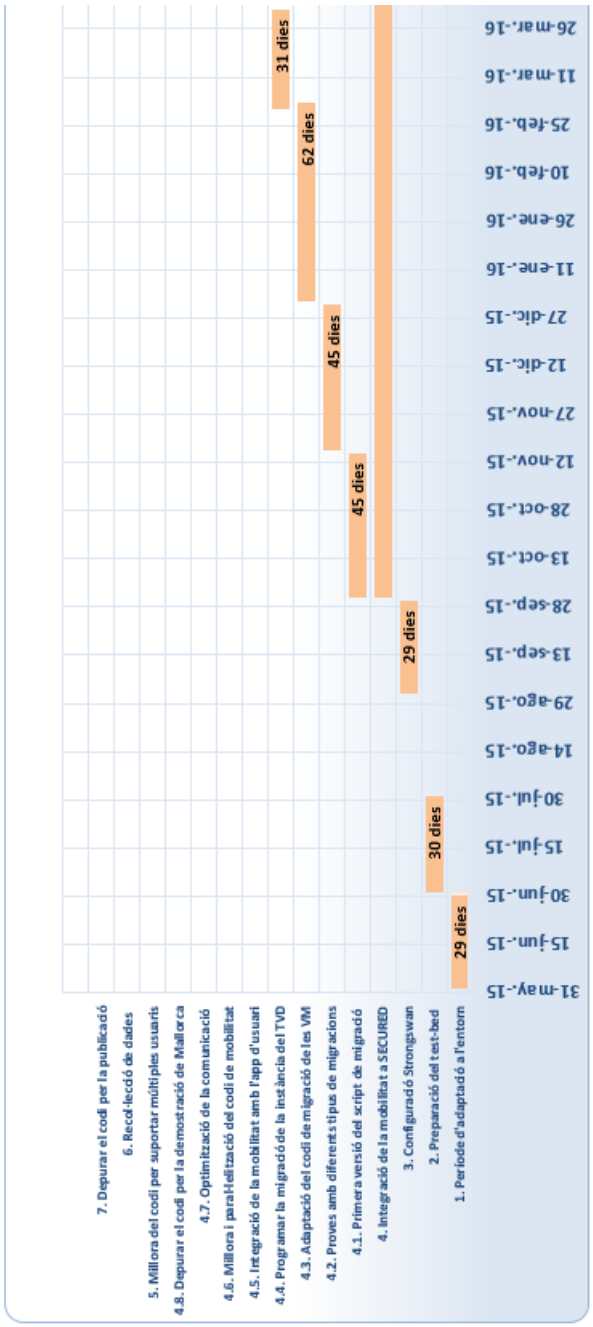


Figure 3.2: 1st part of the gantt diagram

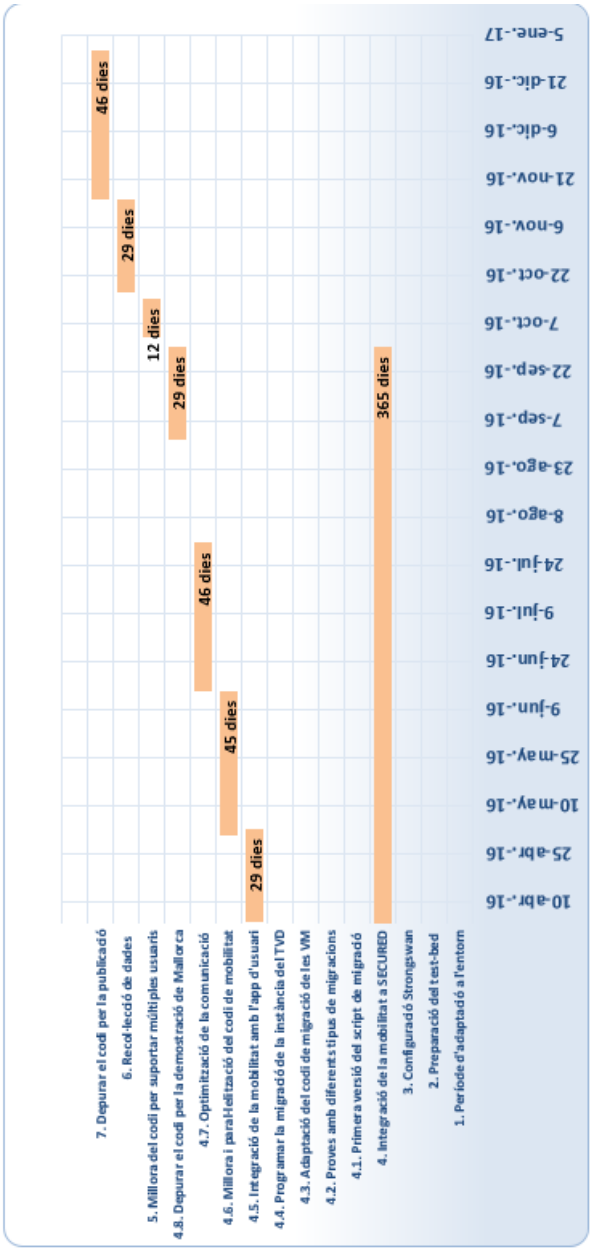


Figure 3.3: 2nd part of the gantt diagram

Chapter 4

Budget

4.1 Budget

In this chapter there is an identification and estimation of costs of the part of mobility of the project SECURED.

4.2 Identification and estimation of costs

Throughout the project the different expenses can be grouped into three different categories:

- Developer
- Hardware
- Indirect costs to the project

4.2.1 Developer

The expenses that are collected here are related to the payment of salaries to workers. The members of the development team participating in this project are:

- The student will assume the role of developer of a part of mobility of the project SECURED, devoting the total of his working day to this.

For the previous participants, it is necessary to consider a series of considerations:

1. For the student, a research contract is agreed upon, destined entirely for salary. In this case, 10800 €will be distributed among the 18 months of project duration.
2. In the case of the assistant, this is involved in the project SECURED, with a remuneration of 1000 €

Considering the above considerations, the staff costs table is as follows:

4.2.2 Hardware

Corresponding to the work tools that are made available to the student so that he can carry out his activity within the company.

The hardware that has been made available to the student is:

Rol	Concept	Amount
Developer		
	Salary	10800€
	Total	**10800€**

Table 4.1: Table of staff costs

- Workstation, with a purchase value of 900€
- Two teams for the tests, in NUC format, with a purchase value of 500 €/ unit

As they are goods bought for the occasion, but which will be used in more projects, the full amount can not be attributed, otherwise the corresponding amortization will be applied after the 18 months the project lasts.

The amortization table is as follows Taking into account the previous table, the amounts

Type of element	Maximum coefficient	Maximum period of years
Electronic equipment	20%	10

Table 4.2: Depreciation table (from 1/6/2015)

attributable to each item are:

- Cost attributable to the work station

$$\frac{900 * 20\%}{18 mesos} = 10€/mes$$

$$10€/mes * 18 months = \mathbf{180€}$$

- Cost attributable to the NUCs

$$\frac{(2 * 500€/NUC) * 20\%}{18 months} = 11.11€/mes$$

$$11.11€/mes * 18 months = \mathbf{200€}$$

4.2.3 Indirect costs

The concept of indirect costs includes all that is not directly related to the project, if not applied to the company in general.

Indirect expenses can be classified as:

- Expenditures in light, water, gas, total 200 € per month.

Therefore, in addition to the costs of the project, we must add an amount of 200 € per month, in terms of indirect costs.

4.2.4 Cost chart

The following table shows, by way of summary, the cost of the project for its 18 months duration:

Concept		**Monthly Cost**	**Total Cost**
Personal			
	Developer	600 €	10800 €
Hardware			
	Work station	10.0 €	180 €
	NUCs	11.11 €	200 €
Infrastructure			
	Several expenses	200 €	3600 €
Total			**14780 €**

Table 4.3: Project cost chart

Chapter 5

Architecture

The purpose of the SECURED project is to provide an innovative architecture for the protection of external threats to devices, through the execution of the most common security applications in a programmable device in the access points to the network. In this chapter, we will describe in detail the overall architecture of the components that make it up, with particular emphasis on the acrshort PSC, acrshort PSCM, PSAs and the acrshort TVD of the user and the NED that encompass all of these elements

In the Figure 5.1 we observe the general photography of the NED with the internal components that will be described next.

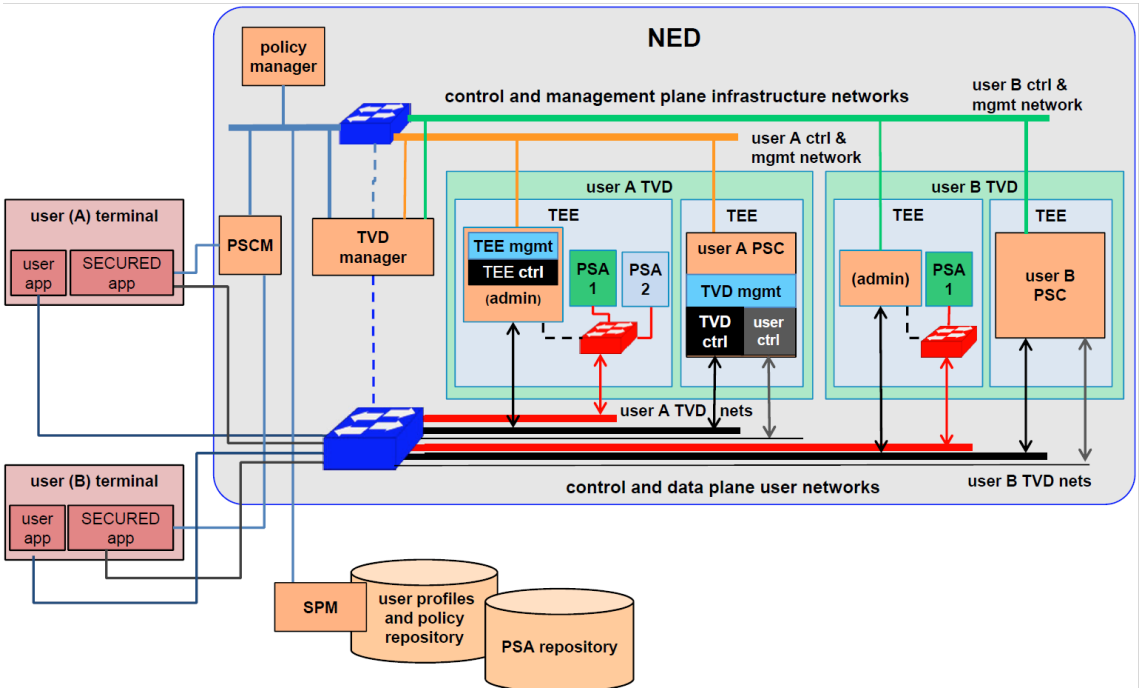


Figure 5.1: NED

5.1 Personal Security Application (PSA)

The Personal Security Application is the atomic entity of the SECURED architecture. It's responsible for enforcing one or multiple security policies for users. Each PSA is composed of a series of security controls. This performs a series of primitive operations oriented to the processing of the network, such as the filtering, segmentation, loading and reassembly of the packages.

A complex security policy can be applied to a single PSA or by connecting multiple PSAs. The chain could be between the PSAs that are executed within the same TEE or between PSAs that are executed in different TEEs but belonging to the same TVD. A series of PSAs correctly connected to the application of a security service implements the beam concept of PSAs.

The PSA architecture internally must be divided into two plans: Control-management and data.

1. **Control & Management plane**: is the part of the software that provides the interfaces to the control and the part of management of the TEE for the configuration, monitoring and requests of signaling from and towards the PSC.

2. **Data plane**: is the part of the PSA that is in charge of the processing in the traffic of entrance/exit of line, providing interface for the layer of communication of the execution environment and allowing interchange of packages between the different announcements of public service and also externally. The internal logic of the PSA can be divided into different modules.

5.1.1 Tipus de PSAs

Antiphising

PSA anti-phishing [10], as its name suggests, attempts to identify the content of impersonation contained in web pages and email or block users from being tricked.

VPN corporate

In PSA VPN corporate, it allows you to connect to your company's network and go to the Internet by emulating that you are in the network of it.

Control Parental

The PSA parental control is the most powerful service for parents to control the devices of their children and prevent them from entering adult and unwanted content sites.

Others

There are more types of PSAs, despite the mobility project, it worked with these three. Therefore we only mention that there are more types of functionalities.

5.2 TEE

Each compartment of a user contains a runtime environment. This environment must be safe and reliable enough to separate competing applications without interfering with each other. Private access is required to manage, configure and start applications. The applications are deployed as we see in the figure, in an area without privileges of the TEE.

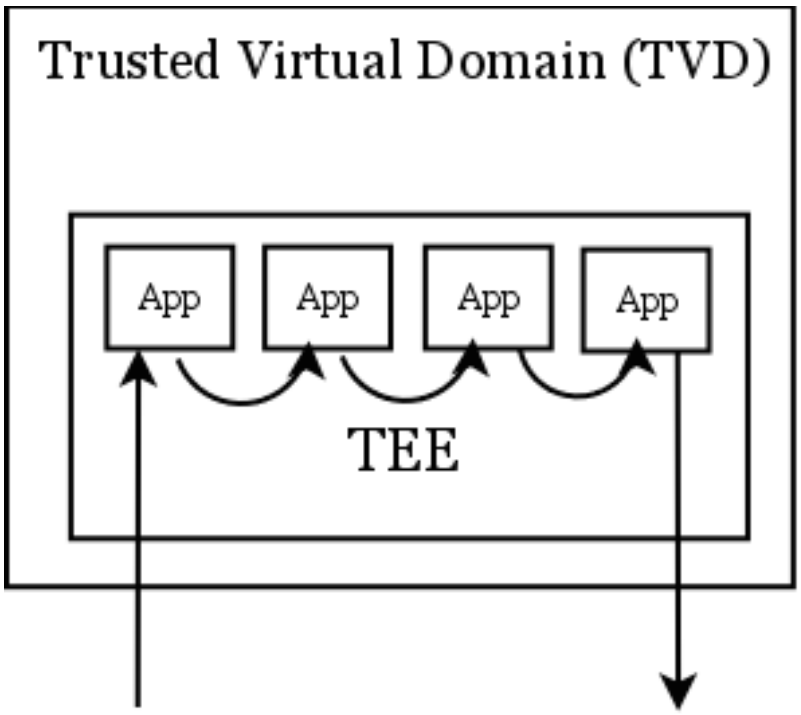


Figure 5.2: NED

5.3 Personal Secured Control (PSC)

The personal security driver (PSC) is the component that controls and manages the TVD, user agents, configurations and network interfaces in the name of a single user. Each user has his own instance of a PSC within his TVD, which captures the entire state of the user and the context of his session. Since the PSC has the privilege of loading applications and enforcing the configurations, because of the design requirements it's isolated from the applications. Therefore, all user-related applications must communicate to their PSC through an interface of Control and Administration restricted.

However, changes in the global TVD (such as the creation of a new TEE) require the sending of a request to the orchestrator, which is an external privileged entity outside the TVD.

Funcions principals:

1. Store the abstract service graph (PSA) and user interconnections;
2. Determines the TVD topology (completion of the service graph) of the PSA requirements (this may change depending on the NED resources);

3. Sends the TVD topology to the TVDM (the latter contains the entire TVD topology and how 1 TVD is connected to another);

4. Oversee the status of the TVD (detect for example if the TEE has crashed and needs to reset);

5. PSC receives the manifests of all public service announcements and is assigned to each PSA to perfection.

5.3.1 Trusted Virtual Domain (TVD)

The Trusted Virtual Domain is a logical abstraction that is not assigned to any physical component of the NED. However, it provides the way to identify all the components that are associated to a user. TVD represents an isolated portion of the NED that is dedicated to a single user and that includes the different execution environments, instances of logical interfaces associated to a user, to implement all the data plane and the chain of the services that it has configured

The user TVD is a logical partition within the host, therefore it can not be assigned to a particular component from the perspective of the implementation. On the other hand, the TVD can be considered as a set of components that belong to a specific user.

5.4 Communication between the elements

The communication between the PSC, PSCM, TVDM and the client is carried out with a REST Gunicorn API.
'Green Unicorn' which is a HTTP server for Python Web Server Gateway Interface for UNIX. The server gunicorn is widely compatible with several web frameworks, simply implemented, low resource consumption of the server, and quite fast.
We will enter in detail about the different APIs that have been modified in the chapter of 6

5.5 Personal Security Controller Manager (PSCM)

The Personal Security Controller Manager is the set of components responsible for communicating the user with the level of control (therefore the traffic the users who will be processed by a PSA does not pass through the PSCM). Internally, it is connected mainly to the control and management network that also connects the NED to the SPM and the repositories.

The PSCM takes advantage of two main components: the remote confirmation agent and the authentication module, which are used in the early stages of the user connection, which are the establishment of a secure connection between the user terminal and the NED, user authentication

Once these steps are completed and user access is granted, the PSCM transfers the user's session record to the TVDM, which will continue the creation of the user TVD instance. Finally, the PSCM reports the initial response of the creation of TVD to the

user after receiving the TVDM.

5.6 Trusted Virtual Domain Manager(TVDM)

The orchestrator is the decision-making component that assigns resources from a user's TVD, creates the connections of this TVD, its connectors and the configuration of its data plane. It doesn't store the user's context, but it must be able to easily retrieve its internal topology in the NED (for example, network configuration and virtual machines that it had previously instantiated).

5.7 Network Edge Device (NED)

This component hosts all the components described in this chapter, and security applications running concurrently, providing a uniform control point for all connected devices. This independent, secure and trusted administrator can act in several different scenarios, such as residential walkways, company routers, mobile points or public access points.

The architecture allows different users (for example individual users, corporate TIC administrators and network providers) to install on demand and execute Personal Security Applications (PSA) within their own trusted and virtualized execution environment. NED hosts and isolates user environments, as well as isolation from traffic flows from different users. Users are able to configure their services through their own Personal Secured Control (PSC) of the NED. Therefore, the security architecture of the NED must be carefully designed to deal with the different needs in a secure way.

In addition to the NED, it should be mentioned that there are also external entities, such as the PSAs (PSAM) manager, the PSA repository (PSAR) and the Security Policy Manager (SPM): they are the centralized services of users download or manage Your contracted services and configured security policies. Users can register with and access these guaranteed services and be able to choose the high level policies they need to be configured.

Chapter 6

Implementation

This chapter will deal with all modifications made to the SECURED project to carry out the implementation of the mobility.

6.1 Instantiation

We will start talking about the instantiation of a user in the environment of SECURED.

6.1.1 Before implementing the migration

At the moment of the development and implementation phase of the mobility of the project of SECURED, we are facing the next scenario. The user can, via a web interface, connect to the NED, create a strongswan tunnel, to create a trusted channel. However, once the possibility of portability and replication of the TVD and the rest of the user environment that has just been instantiated does not exist. You can navigate, in a controlled and secure way, only in the NED that authenticates your user.

6.1.2 NED instantiation steps

Here is a detailed explanation of the main steps of a user to connect to the Internet using a NED:

1. **Establish a trusted channel:** This step consists of two phases: First phase, the user establishes a secure communication with the NED. Second phase, the user verifies the identity of the NED and the user's independent components (PSCM, Administrator of directives, TVDM , NED management, etc.).

2. **User authentication:** NED authenticates the user using their credentials, if authentication is successful, PSCM receives subsequent session signal.

3. **Request for TVD instances:** PSCM asks the TVDM for a basic TVD instance for the user (and the session's testimonial is passed). Initially, the TVD will only include the PSC and will then be expanded depending on the user's graph of service.

4. **PSC Consultation:** TVDM uses the session identifier to search the user profile (PSC for its corresponding section).

5. **Creation of TVD and PSC start:** The TDVM of the NED creates the TVD and the PSC of the user starts.

6. **View user profile:** TVDM retrieves the user profile (that is, the graphical user service graph) of the SPM using the session token.

7. **Translate policies, upload user profile and configuration of PSAs:** If necessary, then just in time, policy setting is carried out by the Policy Administrator. Subsequently, with the infrastructure graph, the configuration of the profile of PSAs and the user is sent to the PSC of the user.

8. **Get public service announcements:** Before requesting the TVD expansion, PSC questions ask TVDM for PSA. The TVDM download PSA from the user using the provided session token. While the PSA manifests are always returned to the PSC, PSAs can also be charged to the latter if they need to be manually loaded into the TEES.

9. **The TVD topology is determined:** the user's PSC the in frastructure graph, configured and manifest PSA to determine the topology required for the instantiation of the user's service.

10. **TVD expansion request:** PSC sends a request for TVD expansion to TVDM to create new interfaces for PSAs. This request is issued through the TVD MGMT interface.

11. **Enlarge TVD:** TVDM creates the TEEs requested in the user's TVD, in accordance with the request for the TVD's expansion of the user.

12. **TEE TVD setup:** PSC will set the tees properly and send them to the network configuration (using the Ctrl/Mgmt interface). This is necessary to properly connect the multiple public service announcements that are executed in the same ETE.

13. **Low level configuration of PSAs:** PSC will charge PSAs and their low level settings in the respective TEE. PSC public service announcements will then start.

14. **Type of report:** PSC reports on its status to the management of the NED and the second is the second stage certification of the PSC and public service announcements. Then NED management generates a global status report and sends it to the PSCM.

15. **User evaluation:** User terminal receives status report from the PSCM, verifies status report is fine, disconnects the PSCM connection.

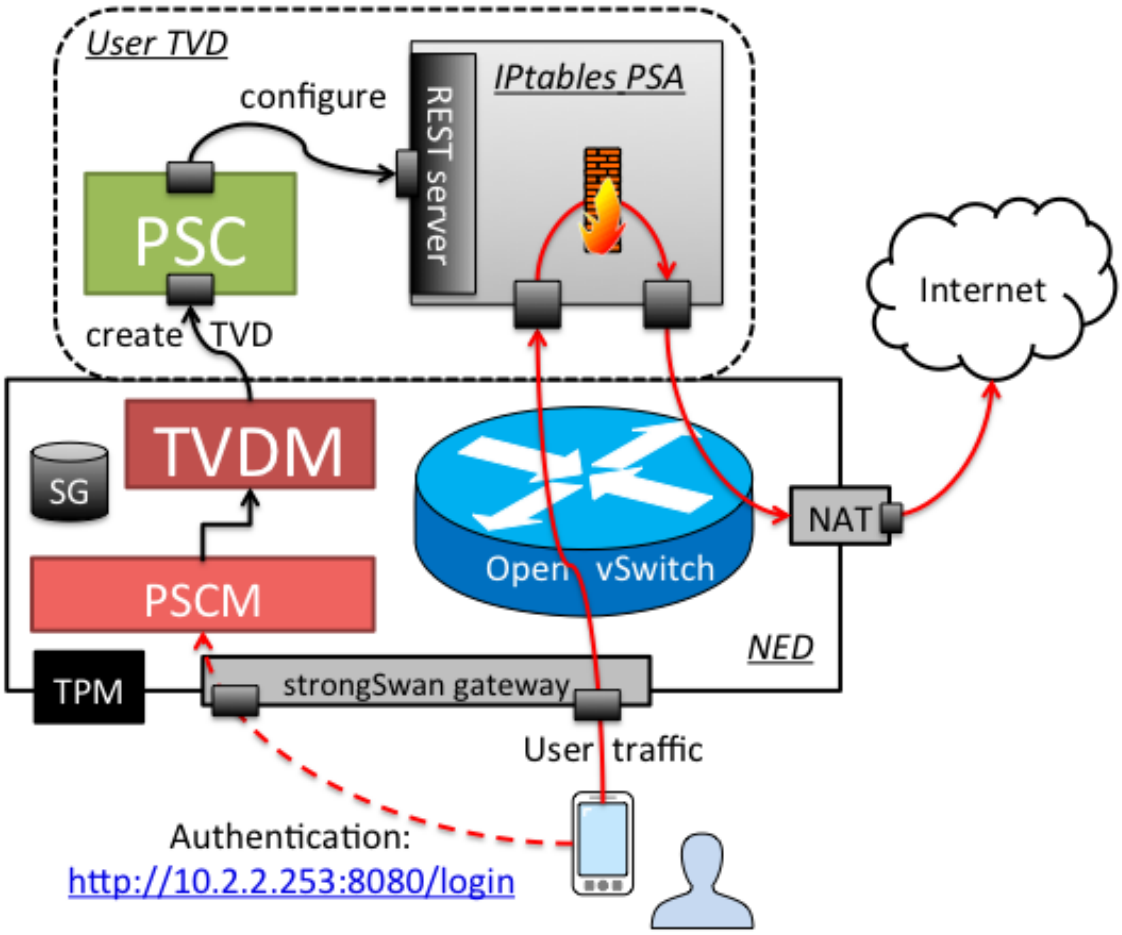


Figure 6.1: Instantiation

6.1.3 Final stage

Once the implementation of the mobility has been completed, everything explained in the previous point, except for point 1 and also adds a new point:

1. Establish a trusted channel: It will now be done with an application that works on the client's computer (implementation by my partner Alicia). Along with all the new implementations in StrongSwan, the application makes the communication with the NED to start the trust channel.
2. All points, minus number 1, described in the previous section.
3. Signal verification: The application looks at all the signals of the NEDs nearby, and sends REST calls to the PSC to indicate that if the migration process is necessary or not.

After explaining in detail all implementations made at the code level and internal functionalities of the NED. We will detail the following functional points of the migration.

6.2 Implementations made for mobility

6.2.1 VPN Tunnel with StrongSwan

First of all the connection had to be safe, it is decided to use StrongSwan, a solution based on virtual protocol network. Version 5.5 was compiled with the following extensions or plugins:

MOBIKE: The MOBIKE[19] IKEv2 extension (RFC 4555) allows an already generated tunnel to change its anchorage point of the network. StrongSwan implements MOBIKE observing the interfaces, addresses and routes. If the configuration changes, route queries are performed to find a better path than the current one and, if necessary, the path is changed by means of a MOBIKE update. strongSwan must switch to the UDP 4500 port from which the MOBIKE extension uses this port for the communication of this port because from the request of IKE_AUTH, which includes a notification MOBIKE_SUPPORTED, although NAT was not detected.

Plugin SQL: The SQL plugin for Charon allows you to store the complete link configuration in a relational database. In addition, the daemon reads the credentials, such as certificates, private keys or passwords in the database to do all kinds of authentication. Logging in to the database is also possible. With these two elements we obtain the way to preserve the tunnel in a change of initial configuration, at the time of the migration of the user's environment and with the mysql plugin we can have a user registry, to support multiple users connected simultaneously in a single NED. We generate users with the ips that will be assigned. Each user will have a 10.2.1.x/32 rank ip.

6.2.2 Mobility functionalities added to the existing classes

mainIPSEC.py A route has been added for the migration functionality of the TVD gunic API.

Code snippet 6.1: New route of TVDM functionality

```
1 app.add_route('/migration', orch)
```

The route that initiated the installation has been modified, with a new parameter. It is the flag of activation of migration. This flag controls whether the entire user environment should be scratched or regenerated

Code snippet 6.2: Added new parameter to the instantiation path

```
1 orch = Orchestrator(instantiator, migration)
```

And the instantiation of migration

Code snippet 6.3: Institution of migration

```
1 migration = TVDMigration(instantiator, conf, logger)
```

Orchestrator.py It's the TVD orchestra, that is to say, the aforementioned TVDM in the chapter 5, is the TVDM gunicorn API. New features have been added. Now when you capture a REST POST call, it starts with client migration with a REST POST call that you receive from the PSC.

Code snippet 6.4: Function on_post()

```
1     def on_post(self, request, response):
2         '''
3         exclusive for migration
4         '''
5         try:
6             self.instantiator.tvdm_receives_migration_request =
                  datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
7             args = request.stream.read()
8             self.TVDMigration.instantiator.logger.info(request.
                  method + " " + request.uri + " " + args)
9             session = json.loads(args, 'utf-8')
10
11            self.eventList[session["token"]] = Event()
12            self.handoverEvents[session["token"]] = Event()
13            migration = Thread(name=session["token"], target=self.
                  TVDMigration.init_migration,
14                               args=(self.eventList[session["token
                                  "]], self.handoverEvents[
                                  session["token"]],), kwargs={"
                                  session": session})
15            migration.start()
16            response.status = falcon.HTTP_200
17        except:
18            self.TVDMigration.instantiator.logger.exception(sys.
                  exc_info()[0])
19            response.status = falcon.HTTP_500
```

Additionally, the functionality that captures REST GET calls to respond to the PSC has been added if you have to notify the client if you can change the access point.

Code snippet 6.5: Function on_get()

```
1     def on_get(self, request, response):
2         '''
3         Exclusive for migration, non-bloking query migration
4         '''
5         try:
6             self.TVDMigration.instantiator.logger.info(request.
                  method + " " + request.uri)
7             user = request.get_param("user")
8             action = request.get_param("action")
9
10            obj = {}
11            if action == "migration":
12                if user in self.eventList.keys():
13                    self.TVDMigration.instantiator.logger.info("
                          user: %s, eventList: %s" % (str(user), str
                          (self.eventList[user].isSet())))
14                    obj = {"status": self.eventList[user].isSet()}
15                    if self.eventList[user].isSet() is True:
16                        self.instantiator.
                              tvdm_sends_migration_finished =
                              datetime.utcnow().strftime("%Y-%m-%d %
                              H:%M:%S.%f")
17                        self.handoverEvents[user].set()
18                    else:
```

```
19                              obj = {"status": False}
20                  else:
21                      obj["timestamps"] = self.instantiator.timestamps
22                  response.body = json.dumps(obj)
23
24                  self.instantiator.logger.info(str(response.body))
25                  response.status = falcon.HTTP_200
26
27              except Exception as e:
28                  self.instantiator.logger.exception(sys.exc_info()[0])
29                  self.instantiator.logger.exception(str(e))
30                  response.status = falcon.HTTP_501
```

Graphinstantiator.py The migration flag, mentioned in the preceding subsections, is added to control whether the installation of the TVD must be performed from scratch or collect the data given for the session that receives the product from the migration to regenerate the policies The PSC's Openflow routing and PSAs of the user, virtual interfaces are to say the entire data plane.

userTVD.py There is a flag of migration, this flag controls whether the TVD configuration should be generated from scratch or given by a migration, so you just have to regenerate interfaces and OpenFlow policies.

In annexes B, commented with###migration code### to indicate the modifications that have been made in the already existing classes.

6.2.3 The new implemented class: TVDMigration

Below we will give a brief explanation of some of the main functions of the new implemented class that is responsible for the migration of the user's environment

SendTVD function

It is responsible for preparing in the file JSON the entire user environment that was generated in the source NED (TVD). Includes names of virtual machines (PSAs and PSCs), names of the virtual interfaces generated, ip assigned to the PSC and the configuration of the "data plane" of the uusari. Also activate the migration flag and also save it in gls JSON. And finally the user's token.

All of this information is entered in a message JSON and sends a REST call to NED destination where this message and destination receives a function that performs the reintanciation of this whole environment.

Code snippet 6.6: Funció SendTVD

```
1     def sendTVD(self, session):
2         '''
3         Obtain user's TVD and migrate this
4         '''
5
6         self.TVD['token'] = self.instantiator.userTVDs[session['
              token']].userName
7         self.TVD['IP'] = session['IP']
8         self.TVD['userInterface'] =
```

```
9           self.instantiator.userTVDs[session['token']].
                userInterface
10          self.instantiator.logger.info("userIsessionnterface " +
11           str(self.TVD['userInterface']))
12          self.TVD['vlanID'] = self.instantiator.userTVDs[session['
                token']].vlanID
13          self.TVD['pscAddr'] =
14          self.instantiator.userTVDs[session['token']].pscAddr
15          self.TVD['psc'] = self.instantiator.userTVDs[session['
                token']].psc
16          self.TVD['pscName'] =
17           self.instantiator.userTVDs[session['token']].psc['name']
18          self.TVD['generatedFlows'] =
19           self.instantiator.userTVDs[session['token']].
                generatedFlows
20          self.TVD['PSAs'] = self.instantiator.userTVDs[session['
                token']].psaList
21          self.TVD['psaIPaddresses'] =
22           self.instantiator.userTVDs[session['token']].
                psaIPaddresses
23          self.TVD['migration'] = "True"
24          self.TVD['action'] = "TVD"
25
26          try:
27              TVD = json.dumps(self.TVD)
28              self.logger.info("TVD: " + str(TVD))
29          except:
30              self.logger.info("Error to created TVD json")
31
32          try:
33              cmd = "ip netns exec orchNet curl -X PUT --header \
34              Accept: application/json --header Content-Type:
                    application/json -d\
35               '" + TVD + "' http://192.168.1.1:8080/migration"
36              (results, errors) = self.execute_command(self.ssh, cmd
                    , False)
37              self.instantiator.tvdm_sends_TVD =
38               datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%
                    S.%f")
39
40          except:
41              self.logger.info(errors)
```

Migration function

Start the migration thread of the PSC and the PSAs, this function will launch different theards that will be controlled by a gls flag to know when to send the message rest with the usurri configuration, which names can be made when all virtual machines have migrated to their destination.

The code part is in the annex B

Obtain_disk_base function

Get the base disk of virtual machines

Code snippet 6.7: generate_remote_disk function

```
1     def obtain_disk_base(self, path):
2         '''
3         obtain path VM disk base
4         '''
5         try:
6             proc = subprocess.Popen("qemu-img info " + path,
7                                     stdout=subprocess.PIPE, shell
                                        =True)
8             (out, err) = proc.communicate()
9             returnedValue = str(out)
10            start = 'file: '
11            end = '\n'
12            disk_base = ((returnedValue.split(start))[1].split(end
                  )[0])
13
14        except subprocess.CalledProcessError as e:
15            self.logger.info("Calledprocerr:" + e)
16
17        return disk_base
```

generate_remote_disk function

Check first that there is a virtual machine disk in your destination but creates it and migrates it. This solution was carried out to see that it depends on what scenarios the migration of the disk was giving error. This did not entail either a penalty in time since the virtual machine disk was very small.

Code snippet 6.8: Generate_remote_disk function

```
1     def generate_remote_disk(self, original, disk_type, path, vm):
2         """
3         Generate remote disk
4         """
5         stderr = ""
6
7         try:
8             cmd = "[  -f " + path + "/" + vm + " ] || qemu-img
                  create -b "
9                   + original + " -f " + disk_type + " " + path
10            stdin, stdout, stderr = self.ssh.exec_command(cmd)
11            stdin.close()
12        except Exception as e:
13            self.logger.info("Error to create remote disk " + str(
                  e) + " "
14                             + str(stderr))
```

Rename_remote_disk function

If the previous function generated the remote hard disk, it checks that the name is the one of the machine that will be migrated.

Code snippet 6.9: Rename_remoter_disk

```
1     def rename_remote_disk(self, path, vm):
2
3         stderr = ""
4
5         try:
6             cmd = "mv " + path + "/" + vm + " " + path + "/" + vm
                  + "backup"
7             stdin, stdout, stderr = self.ssh.exec_command(cmd)
8             stdin.close()
9
10        except Exception as e:
11            self.logger.info("Error to create remote disk "
12                             + str(e) + " " + str(stderr))
```

CreateSSHclient function

The function implemented makes you the paramiki library, for the creation of an SSHClient object. For more direct control, it uses a socket (or socket-like object) for transport, and uses start_server or start_client to negotiate with the remote host as a server or client. In this way the NEDs are communicated between them to start, maintain and end the migration of virtual machines.

Code snippet 6.10: Funció createSSHclient

```
1     def createSSHClient(self, server, port, user, password):
2
3         client = paramiko.SSHClient()
4         client.load_system_host_keys()
5         client.set_missing_host_key_policy(paramiko.AutoAddPolicy
              ())
6         client.connect(server, port, user, password)
7         return client
```

TVDMigration super class

This class that executes the migration of virtual machines (PSASs and PSC). With the use of the libvirt library, the connection with the remote NED begins, by using the socket created with the function createdSSHclient, explained above, once it is established, start the hot migration of the virtual machine. Once finished, it removes the machine that has remained in standby, to avoid excessive consumption of memory.

The code part of the super class is in the annex B

6.2.4 New features included in the REST gunicorns API

As explained in the chapter 5, the NED is running multiple API REST gunicorns. To implement the migration a series of new features was implemented. The communication between the client and the server was also done by capturing REST JSON messages.

Mobility extensions to the NED basically cover three different parts, and) migration interfaces of virtual machines in the PSCM, ii) of the network status migration API on TVDM and iii) of the API of signal reports to the PSC.

The main flow of migration work is as follows:

1. The customer informs the Wi-Fi signals of the access points. PSC evaluates the signal status, which is correlated with the past. In case you do not need to migrate, the system returns.

2. If migration is necessary, the system invokes the migration process of virtual machines within the PSCM.

3. At the same time as they request the migration of the status of the TVDM network.

4. The mobile node is informed to shoot the tunnel change.

Changes to the TVDM API The TVD migration is based on the replication of the network configuration in the destination system add-ons made in the TVDM, which provides the necessary logic for the migration of the TVDM to the other NED. To this end, we add a new call to the API: migration, which assumes in the TVDM migration, and another one must capture the migration, *instantiateTVD*, which provides re-instantiation capabilities of a TVD. The detailed explanation of the added code is in the subsection **??**.

Changes in the API for the PSC Finally, to allow the client to provide information about the signal strength of the mobile node we have provided a new method that allows these reports, and also provides a suitable response to the client with two options:

- FORCE_HANDOVER: It allows the client to complete the migration, as necessary, as soon as possible.

- NO_HANDOVER: Blocking option, so that the client must remain associated at the same access point.

6.2.5 Parallelization and flow control of the execution of mobility

As explained in the chapter 3. Once it proved that the integration of the migration worked, the optimization of this one was carried out. The solution chosen, with the Events class of the Threaring library. When calling an Event(), this function returns a new Event. An event is managed by a gls flag, by default it is false. And it is set to true with the Event.set() method. The Event.wait() method, a thread is blocked until the flag is true.

To optimize migration, we use events to control the flow of mobility execution. As already seen above, threads are already used to start the migration. However the thread is unique to each user and is iterative. Now an event is incorporated to know when the migration of PSAs ends and to pass the customer testimony to make the change of NED. Therefore, once the event has been created, we keep a list of events hash, the key will be the user's testimonial. Once saved, a migration instance is generated, where we pass the session parameters (where the user's testimonial is located) and the event.

Within the migration instance of each user, the migration of all PSAs is started in parallel: each one will generate an event, which will be saved in the list. After launching all migration threads of PSAs.

A thread was prepared to migrate the PSC, it will not begin to migrate until all PSAs have not migrated. It will also create a user's TVD thread. This thread will be blocked waiting for the migration of the PSC to be started, then the TVD will be sent.
While the primary thread of migration is running, the PSC is launching REST calls on_get(), this process is repeated, until the initial event is not put to true. Each of these PSAs also has an event, therefore When all the events of the PSAs are true, the main event is unlocked. At this point the PSC is unlocked to send the confirmation message of change of access point to the client.

All threads will end and the memory space is freed with a general join(). To ensure that no thread is open.

6.3 Final stage: Migration

1. **PSC captures client migration message:** The PSC captures the REST migration start call, passes the testimony to the TVDM with a variable session that contains among other things the user's key and the ip of the NED where it must migrate.

2. **TVDM captures the message from the PSC:** The TVDM hunter receives a message REST on_post () with the session variable, generates the user migration event and launches the migration thread.

3. **Migration PSAs:** First all the PSAs of the user begin to migrate

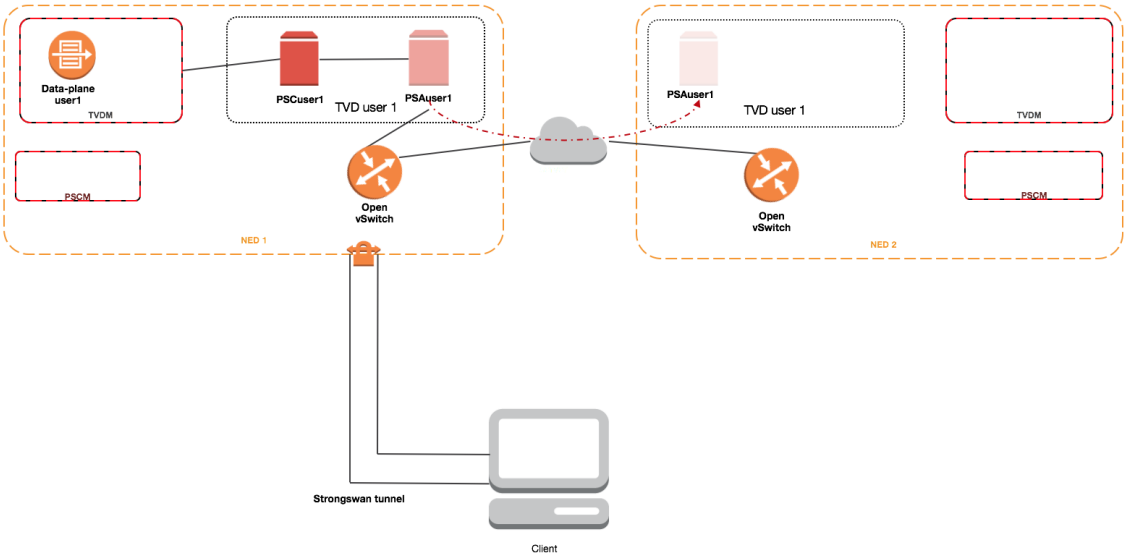


Figure 6.2: Migration of PSAs

4. **PSC migration:** When the migration of all PSAs is at 80%, the migration of the PSC begins

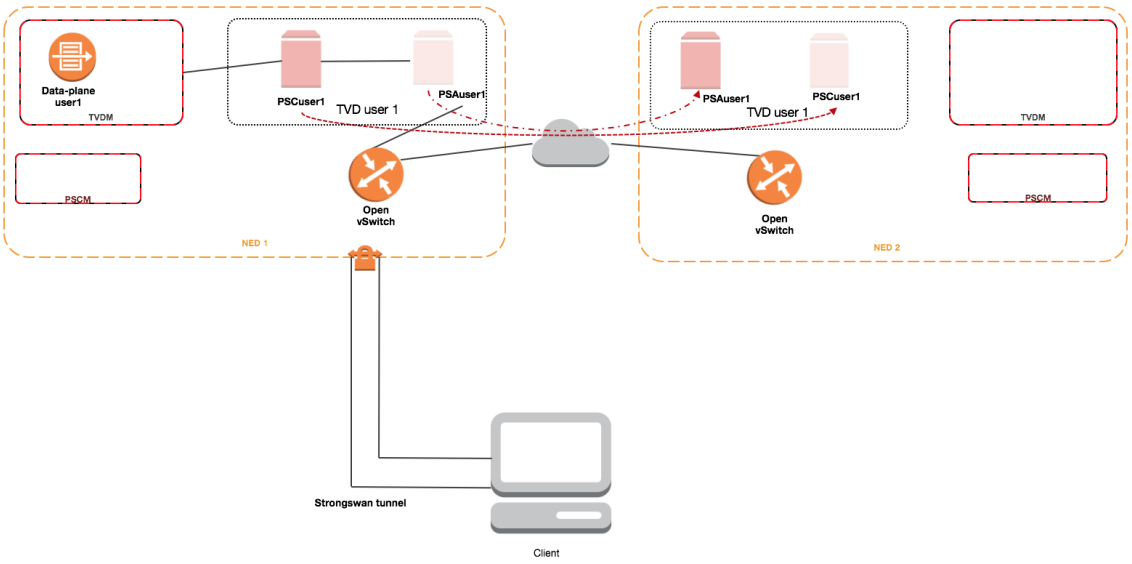


Figure 6.3: Migration of PSC

5. **TVDM responds to the PSC:** The PSC once the migration has begun, sends REST calls to on_get () to receive a positive response from the TVDM. When the TVDM responds, the PSC collects the testimonial.

6. **PSC responds to the client:** The PSC responds to the client that he can make the change

7. **TVD Shipping:** Instant after the PSC migration begins, it is sent with a REST message, a json with all the user's TVD to be re-instantiated at the destination.

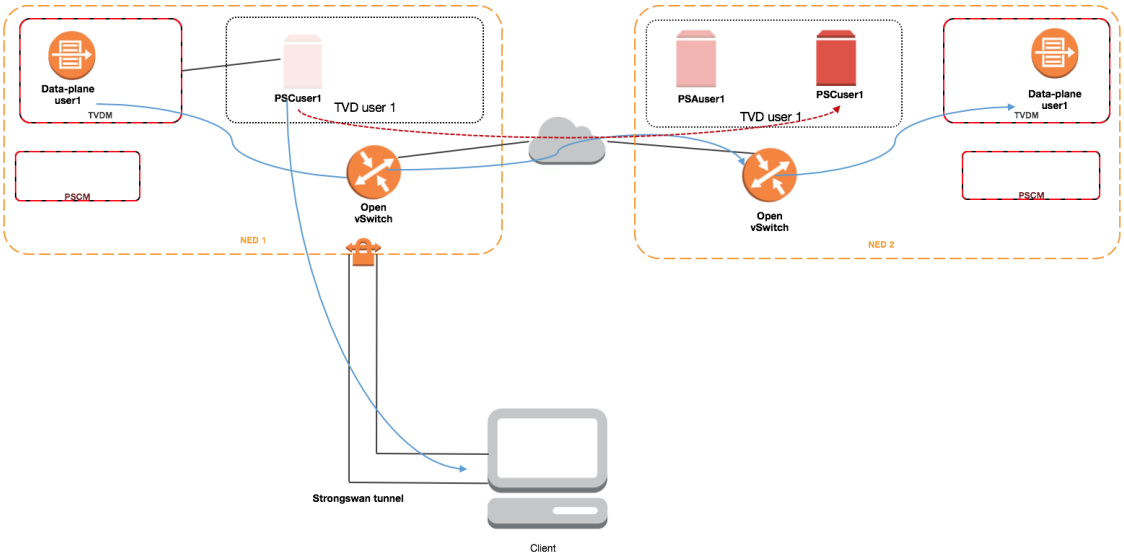


Figure 6.4: Shipping of the TVD

8. **Re-instantiation of the entire environment:** The entire user's data-plane is re-instated, all user interfaces and their configuration are regenerated.

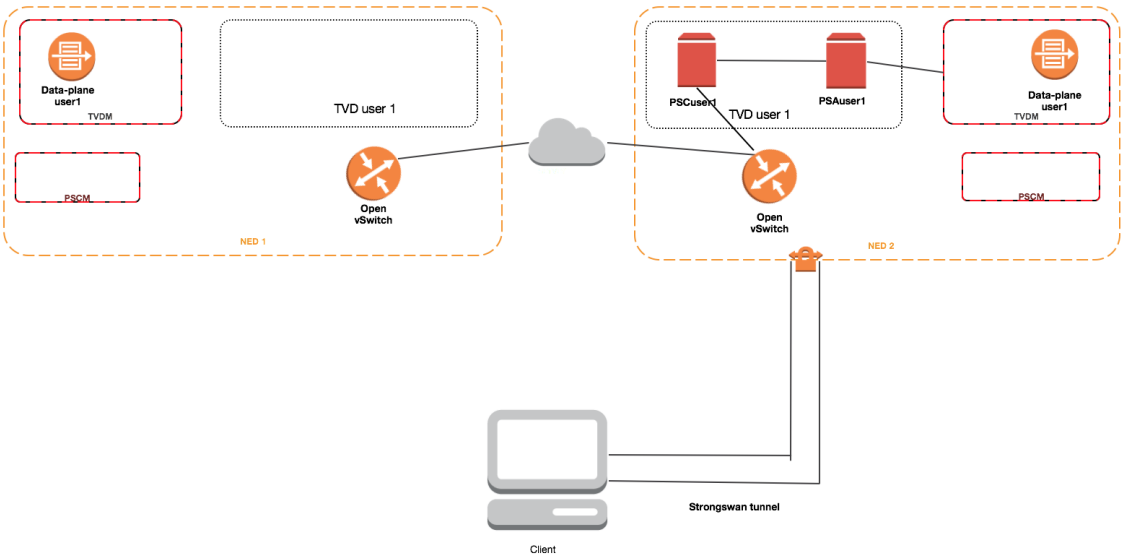


Figure 6.5: Re-instantiation of TVD

9. **The client changes from AP and the end of the tunnel:** The customer drops the StrongSwan tunnel from the NEd source, connects to the new access point and raises the tunnel in the destination NED.

Chapter 7

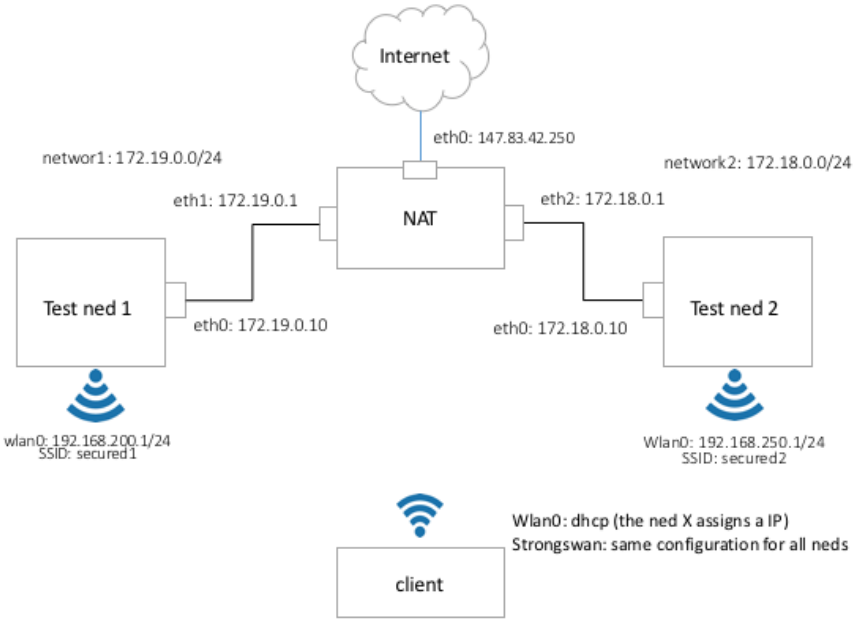
Performance tests

In this chapter we will discuss the environment that has been configured to perform the performance tests of the mobility implementation carried out in the SECURED project.

7.1 Test bench

The configured environment consists of two computers in NUC[17] and a portable format, the two nodes are connected to each other by a local network, with a single Internet connection, as we saw in the figure 7.1 How is a conceptual design.

Figure 7.1: Conceptual design of the test bench



Since the NAT element is nothing more than a gunicorn running on the ned 1 test. This machine is physically the only one that leaves you on the Internet. Therefore every time the change of NED is made, you receive a request with the information necessary to find out where to re-direct transit to a ned or another. This solution was taken to save resources and Add a third physical machine, with the economic and energy costs involved, only to perform this function. Finally we have the following table with the ba-

sic hardware and software information installed on the machines that are used for the tests.

Equipment			
Name	test ned 1	test ned 2	client
System/manufacturer	NUC5i5RYBAF	NUC5i5RYBAF	ASUS
CPU	Intel Core i5-5250U	Intel Core i5-5250U	Intel Core i5-2430M
Memory	8GB	8GB	4GB
Distribution	Debian GNU/Linux 8		
kernel	3.16.0-4-amd64	3.16.0-4-amd64	4.7.0-1-686-pae

Table 7.1: Specifications of the equipment used

7.2 Functionalities added for the performance of the test

The datetime library has been added, making use of the feature:

```
1 |     datetime.utcnow().strftime(\"\%Y-\%m-\%d \%H:\%M:\%S.\%f\")
```

Where we return this date with this format: 2016-11-11 09:37:50.391198

With this function we have compiled the exact time of the actions of the elements that intervene in the migration, without noticing any penalty of performance.
Therefore timestamps have been added throughout the execution stream, since the client reports that he must migrate until the StrongSwan tunnel has been recreated at destination:

1. CLIENT SENT REPORT: Instant of time when the client sends a message to the PSC that must initiate the migration process
2. PSC RECEIVED REPORT: Instant of time in which PSC receives the client's message.
3. PSC SENT MIGRATION REQUEST: Instant after the capture the client's message sends the migration start request to the TVDM.
4. TVDM RECEIVED MIGRATION REQUEST: Time Instant on TVDM receives the migration request.
5. TDVM PSA INI / FIN: The following brands are performed in parallel, for so many PSAs the user has instantiated.
   (a) TVDM PSA INI: Instant of time in which it starts with the migration of n PSA.
   (b) TVDM PSA FIN: Instant of time at which the migration of n PSA ends.
6. TVDM PSC INI: Instantly, the PSC began its migration to the destination NED
7. TVDM SENDS TVD: Instant of time in which the TVDM sends the TVD of the user

8. TVDM PSC FIN: Instantly, the PSC has finished its migration to the destination NED

9. TVDM FINISHED MIGRATION: Instant of time that the entire migration process has ended by the origin NED

10. TVDM SENT MIGRATION FINISHED: Time Instant on TVDM sends an ending message

11. TVDM2 RECEIVED TVD: Instant of time when the TVDM of the destination NED receives the TVD that has sent the TVD of the NED origin.

12. TVDM2 FINISHED INSTANTIATION PSAs: Instant of time in which you have just re-instantiated all PSAs in the destination NED.

13. TVDM2 FINISHED INSTANTIATION PSC: Instant of time in which the PSC finishes re-instantiating itself in the destination NED.

14. PSC RECEIVED MIGRATION FINISHED: Instant of time in which the PSC re-instantiated receives the message of completion of the migration, previously sent by the TVDM of the NED origin.

15. PSC SENT HANDOVER ORDER: Instantly, just after receiving the end of the migration message, send the order to the customer to do the "handover".

16. CLIENT RECEIVED HANDOVER ORDER: Instant where you send the order to the customer to do the "handover".

17. CLIENT CLOSED TUNNEL: Instant where the customer removes the tunnel with the source NED.

18. CHANGED CLIENT APS: Instant where the customer is making the change of access point (it is disconnected from the source NED and is connected to the destination NED).

19. CLIENT CREATED TUNNEL: Instant where the customer raises the tunnel with the destination NED. And here is the entire migration process.

7.3 Executions

The test that has been done has been done with a user with two PSAs instantiated, one with the corporate VPN service and the second one with the firewall service. The collection of data has been initiated when a total of 5 migrations were carried out between the two NEDs.

Each time stamp, calls the function already shown above and then the client collects these marks and generates a resulting file with all the data. Multiple executions have been performed, in the course of a month, and groups of 5 have been selected, to get the data loyalty to the fullest possible and we have no outlier that could differ greatly. The resulting files can be seen in the Annexes.

In the next chapter we will show the results and analyze them.

Chapter 8

Results

8.1 Graph of results

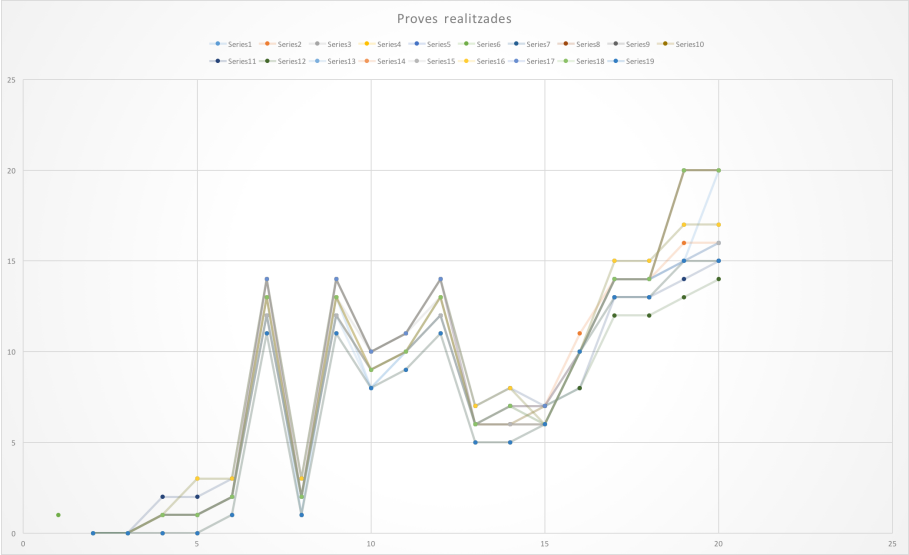


Figure 8.1: Results of the tests

The graph is all tests carried out, in the axis of the X we have the time and the axis Y are the timestamps that have been placed in the code, are described in the chapter 7

8.2 Interpretation of results

As we can see in the results, we can ensure that the maximum migration time is about 17 seconds. If you scale the amounts of PSAs it would affect minimally, as it is carried out in parallel. All the tests were done while the client was watching a YouTube video.
The video has not been cut at any time, the mobility has been operating all the time and the laptop was deliberately moving across the area where the test bench was to force the migration.

Chapter 9

Sustainability

In this chapter you will see the sustainability report. What it intends to show is an analysis of all the effects and expenses that the SECURED project will generate before, during and after its development. How they will affect social, economic and environmental fields so that they can be valued and considered within the project.

9.1 We know that sustainability is one of Sustainability Analysis

We know that sustainability is one of the main challenges of this century. We are aware of the negative effects our actions have on the environment, we know of the existence of the limits that the planet has and the social injustice that is committed every day, as well as the importance of working in a sustainable way. Even so, the concept of sustainability remains an outstanding matter, difficult to define and narrow down.

For this reason, in order to carry out a correct sustainability study of the SECURED project, it has been decided to use an array of scores that are obtained from a list of questions that correspond to each part.

The analysis carried out with this matrix**??** consists of three parts: the start-up part of the project, the useful life part and the results it has and, finally, the risks that may appear. At the same time, each one of them is analyzed from the three dimensions of sustainability: the economic, the social and the environmental.

In the start up part of the project, the part that belongs to the UPC team, looking at the three dimensions, what is sought is to know if the impact of the project was considered, in the section of mobility. For the beneficiaries, for those who manage it, for the planet and also for our budgetary constraint. Know if it has been estimated correctly and the cost that may be quantified has been quantified and if the necessary measures have been taken to reduce this impact.

In the useful life part of the project we can akcnowledge how the problem is solved (as described well in the Reach chapter) and how the current situation will improve the implementation of our solution. We also want to know the impact that will be produced during its useful life and what type of impact the project may have at the time of its dismantling.

Finally, in the part of risks what is wanted is to enumerate those scenarios that are factors of danger to increase the negative impacts that the achievement of the project can produce and be able to have a plan of action or to know how they could be mitigated, both in the environmental dimension with the ecological footprint and in the economic dimension with the viability of the project or in the social dimension to the detriment of some particular sector of the population or the creation of dependencies.

And here we have the sustainability matrix of a project, of which we speak so much, with its range of possible scores included:

	Ambiental	Economic	Social
Project in production	Resource analysis	Economic feasibility	Personal impact
Evaluation:	[0:10]	[0:10]	[0:10]
Lifetime and results	Ecological footprint	Final cost	Impacte social
Evaluation:	[0:20]	[0:20]	[0:20]
Risk	Environmental damages	Economic risks	Social damages
Evaluation:	[-20:0]	[-20:0]	[-20:0]
Evaluation:	[-60:90]		

Table 9.1: Matrix of sustainability and possible scores

Scores are obtained based on the reflections and answers generated by the battery of questions that correspond to each square. This matrix has been designed so that it can be used in any engineering project. Your responses give rise to the sustainability report of the project.

Any person responsible for a project must know and analyze its project in depth, know where it is framed and the effects derived from its implantation and development. We believe that this analysis has an important weight in the project and that it is important to emphasize the sustainability of the project within the framework of development cooperation.

From here, a discussion will be held with the intention of giving the appropriate justifications and being able to score correctly for each square of the array. It will be done for each defined part, the putting in production of the project, the useful life and the possible risks and will end with a final evaluation and the s of this analysis.

To design a sustainable project, a global vision that includes the three dimensions of sustainability (environmental, social and economic) must be applied, and where the relationships between the three components are visible. For that, the dimensions and their interrelations will be analyzed in a common way in the three parts.

9.2 Project in production

The questions you want to respond to in order to carry out the sustainability approach to the implementation of the project are the ones that are presented below:

Environmental Dimension:

- Has the environmental impact of the project been estimated? Can it be minimized by using resources?
- Is the environmental impact quantified? What measures have been taken to reduce it?
- What is the origin of the raw materials used? Is its origin or manufacture ethical?
- Has the dismantling of once the life of the project has been considered?

Economic dimension:

- Has the cost of realizing the project been estimated?
- Has this cost been quantified? What decisions have been made to reduce this cost?
- Has it been adjusted to the expected cost? The initial investment will allow it to be competitive.

Social dimension:

- What contributions do you have to carry out the project on a personal level?
- Has it involved significant reflections on a personal, professional or ethical level in the people who have intervened?
- What is the social and political situation in the country where it will be implemented?
- Will the activity contribute or worsen this situation?

Every engineering project is born of the need to solve a problem or to materialize a solution. The environment of SECURED that is to be generated, must be robust, economically viable, must take into account the easy access and use of the users and must have a sustainable environmental cost. Any project that must be carried out must be subject to a feasibility study.

Regarding the SECURED project, after having carried out an exhaustive evaluation of the material and human resources in the Budget Chapter, we have an estimated figure. This, it lacks sense if we do not relate these resources to the real impact they have on the environment. But first of all, we must bear in mind that this is a cooperation project between different universities and telecommunication companies and that has been financed by the European Union, despite each team receiving a varied budget, the UPC team is of the month modest One of the main objectives has been that the economic expenses were as minimal as possible.

Given this situation, it should be taken into account that the estimated cost that has been presented is not a real economic cost since much of this expenditure has been made in the form of an investment, either in working hours And dedication, in the availability to travel or in the form of donation of disused computer equipment by the team that have left their sand dune by providing a laptop or various components.

It is very important that the **technology** provided by the tests is new, with the exception of the laptop that has been used as a client. But once this has been completed, it can be replicated in infrastructures already in operation. This reduces the negative impact

considerably in the three dimensions. Socially, the beneficiaries will not hinder the change of way of connecting to the network. Economically there is a small investment in human resources at the time of search and development, but eliminates almost all economic investment. Most of the material has been paid for use in this project, although it will be used to many others, once it is finished. And finally, the electrical consumption during its use.

Regarding the **environmental** issue, we could devote a whole chapter to the positive impact of the environment, but we wanted to summarize it by mentioning the most important. The fact that the environment is fully validated, with the use of current computers, without having to buy more devices. This greatly reduces the impact of choosing a different solution: the great costs and resources necessary for the construction of new technology are eliminated.

In short, when we decide that the solution is a virtualized environment. We make a contribution to reduce our ecological footprint and, therefore, we do not diminish the quality or solvency of the solution found.

Socially, it will have a great positive impact because, although it is used by some small group, the security that will provide the environment of SECURED will help protect sensitive data from possible or corrupted by the infection of a virus or the possible subtraction of this, we will be helping all these people to be able to connect safely.

Adapt yourself All these solutions and more reflections have been generated during the project development process. This is the origin of the personal impact that I have received since we have started supporting the architecture and mobility of the SECURED project. This project is for research and cooperation, it places a very different environment to that of a company for lucrative purposes, so the learning curve is high. Until development is over, you don't see how the results could positively affect society. Especially what has been learned more, is the need to know how to adapt to different circumstances to do, even if they are not of your pleasure and any unforeseen event that has occurred. The impact that I am giving are all the knowledge I have obtained and the learning in and out of the race, they could not have a better scenario.

9.3 Lifetime and results

The questions that are tried to answer for the useful life of the project are the following:

Environmental Dimension:

- How is the problem resolved at the moment? In what will environmentally improve our solution of the existing ones?
- What resources will be used during the life of the project? What will be the impact of these?
- Will the project reduce the use of resources? Overall, will it improve or worsen the ecological footprint?

Economic dimension:

- How is the problem resolved at the moment? In what will economically improve your solution?
- What estimated cost will the project have for life? Could this cost be reduced to make it more viable?
- Has the cost of possible updates been taken into account during the life of the project?

Social Dimension:

- How is the problem resolved at the moment? In what will socially improve your solution?
- Is there a real need for the project?
- Who will benefit from the use of the project? Can any group be harmed?
- How does the problem solve the problem initially?

As mentioned in the previous section, this project arises from anxieties of providing, with one of our surroundings, a secure way to connect to the network. To accomplish this, a team must support the new environment, so it is possible to reuse what is available or use one that is already in the market and reuse it.

The solutions proposed with the project SECURED contemplate a future saving in time and money. So that once the solution is configured, the beneficiaries only, through the application will connect to the network through our solution and will not have to worry about anything else. It will not be necessary to make more investments in the time that can be in operation.

The environmental impact will also be reduced to a minimum, because despite having a small energy expense for the use of only one physical equipment, the emission of greenhouse gases is considerably reduced. The fact that the computer can be reused, if it allows it, extends the life cycle of the computer. If this is not the case, the investment is minimal and can be already manufactured and can be reused. Folding the time of use we are saving the energy and resources necessary for the manufacture and commercialization of new equipment. This means that also the creation of electronic waste, highly toxic, is avoided.

The social impact can be measured with the contribution of security and tranquility of the beneficiaries in safe navigation, making use of our solution.

9.4 Risks

For the last block of risks we present the battery of questions that you want to answer below:

Environmental Dimension:

- Could scenarios that could increase the ecological footprint of the project? Is it possible to prevent or mitigate the impact of these possible scenarios?

Economic dimension:

- Can scenarios that adversely affect the viability of the project? Can the impact of these possible scenarios be prevented?

Social Dimension:

- Are scenarios that could cause the project to harm somewhere in the population? Could it prevent or mitigate this impact of possible scenarios?
- Could you create the project any type of dependency that left users in a weak position?

Every project has inherent risks that may appear, because there are many factors that influence and many situations we have to adapt during its possible introduction in a real environment.

In the economic dimension one of the risks that would cause a more important impact would be that you have to buy a new equipment to incorporate the NED, since the equipment that is available is not compatible. It would involve an investment in the adaptation of the new Also there could be other risks that would jeopardize the viability of the project, but the fact that this cooperation project is designed with the will to re-use resources gives a lot of output to the need to adapt to new situations.

In the most social dimension, the beneficiaries, both companies and individuals, should be a telephone operator that is responsible for offering this service. A dependency, which remains in the hands of the operators that work on the project, should assess if it is finally profitable to introduce this system into a package of solutions that can be offered to their clients.

And environmentally, there is a risk that operators will be completely disconnected from the equipment, if applicable, where the proposed solution is working. Once they stop supporting, customers will throw the equipment into the trash. Where they can contaminate instead of following the process of return and recycling. For the rest of the objectives, there is no obvious danger, since if you do not use a physical computer, the environment can be virtualized on computers already in use.

9.5 Evaluation of sustainability

Once the matrix is presented, you can see the questions and make the discussion, we only need to evaluate it and put a note that is the summary of the reasons so that this project can be considered as a sustainable project. Next we teach the completed matrix:

	Ambiental	Econòmic	Social
Project in production	Resource analysis	Economic feasibility	Personal impact
Evaluation:	9 [0:10]	8 [0:10]	8 [0:10]
Lifetime and results	Ecological footprint	Final cost	Social impact
Evaluation:	15 [0:20]	13 [0:20]	12 [0:20]
Risks	Perjudicis ambientals	Economic risks	Social damages
Evaluation:	-6 [-20:0]	-6 [-20:0]	-15 [-20:0]
Evaluation:	40 [-60:90]		

Table 9.2: Matrix of sustainability and possible scores

The scores that have been given in each square are the result of the evaluations discussed in the preceding sections, putting on the balance the effects and costs, both positive and negative to consider of the influences that affect the project in that dimension. To begin, in the production project section, within a score range ranging from 0 to 10, all dimensions have obtained very high scores. In general, this is because, as a cooperation project, the social sphere here becomes more of an objective than only a dimension to evaluate, because the costs and the ecological footprint will not make sense if it is not achieved, That social objective of cooperation.

For this reason we conclude with the following punctuation:

The environmental dimension obtains 9, the economic one 8 and the social one 8.

The following block of project life may have evaluations ranging from 0 to 20 points. We see that, in our case, all dimensions have a positive assessment, because we believe that the resolution of our project has had a real impact on the initial approach and how it was solved earlier. The solutions proposed with our project contemplate future savings in time and money and do not imply excessive maintenance costs. The environmental impact is reduced to the least expression and is taken into account from the beginning, the possible dismantling of the project and its materials. And with regard to the social part, providing a certain amount of time to maintain the beneficiary's browsing environment, prevents the loss of data or theft thereof. It can be said that it will have a positive impact if it is achieved. For this reason we conclude with the following punctuation:

The environmental dimension obtains 15 out of 20, the economic one 13 and the social one 14.

In the last block of risks you can only have negative ratings, since any risk is a danger that can adversely affect the project. There are important risks, and all of them have to be taken into account and see what can be too big losses, if they appear.

There is a risk that unforeseen events may arise that entail great economic costs, or the social risk, which is the most important, that can not be introduced correctly or not a solution that pleases potential beneficiaries. Finally, the environmental risk is present if all the physical machines used are finally replaced by a new solution, they will be demolished and will generate a whole series of 'polluting elements' for the environment. For this reason we conclude with the following punctuation:

The environmental dimension obtains -6 over -20, the economic one -6 and the social one -15.

Finally, we summarized saying that for this project the score was 40 out of 90. We consider this score to be a good figure, although social risk is high. We conclude that the sustainability analysis defines the project SECURED is sustainable.

9.6 Conclusion

A co-operative research project attempts to improve certain social and economic aspects that negatively affect society. For this premise, we look at whether it is feasible to carry out a solution to bring about both positive and negative impact in a real environment.
It has been shown, in the case of Mallorca, that it is possible to adapt the new system to the material resources that an educational center has to offer, without neglecting to reduce the impact of the ecological, social and economic footprint so that they minimize to the possible minimum in a new environment where it has been installed. We have achieved very well the objectives that were foreseen could be those that had the greatest impact, such as the provision of technological infrastructure. Not only has the objective been achieved but also has had a positive effect, because the technological life cycle of

the parcel has increased and with the virtualization of the environment, which implies a reduction of economic, environmental cost. And the dismantling of the demonstration was minimal.

Chapter 10

Conclusion

10.1 Conclusion

Finally we reach the conclusion of the work carried out. Personally it has been interesting to work on a project of cooperation at European level, a unique opportunity to polish and develop tools for group work and personal management.
Until the objectives have been completed, you have to live with the pressure of fulfilling the delivery deadlines and the need to know how to adapt to different circumstances, ways to do it and any unforeseen event.
This has been one of the learnings, but the main impact that this project has had in me is the opportunity to put into practice all the knowledge I have gained in and out of the degree, which could not be better implemented in the real world.
The application of the contents learned in the project has been the most enriching of all of this, as well as learning to solve the different difficulties that were appearing.

10.1.1 Work Achieved

The project has culminated with the objective of creating the initially proposed environment, with mobility included. As we have seen in the chapter 8, the estimated times are between 17 or 20 seconds of jump between NED and NED. They are relatively low times, considering the whole environment has to regenerate in destiny. The tests carried out at the S'Arenal education center, in the municipality of Platja de Palma, were a success. Therefore, we believe that it can be put into operation in any scenario, with minimal penalty for adaptation to the new environment, a minimum cost, as mentioned in the chapter on 9 and adding the security factor.

10.1.2 Work for the future

The project is over and many of the objectives have been met. However you can always take over the work done and continue from this point. For example, a possible integration with NED working at the integration level of Wi-Fi networks with those of 3G/4G. To give the possibility of total portability. At the moment, as we have already mentioned before, we are limited in local environments.
Another line of work would be to move the environment to a docker, or similar technology, to further reduce the size of virtual machines. Currently, the status of the machine and the hard drive are migrated. The docker would reduce the volume of data to move between the NEDs, thus facilitating the possible integration of the jump to mobile telephone networks.

The SECURED project made the decision to work with kvm/qemu for the ease of adapting the architecture of SECURED and avoiding more complexities due to time limitations.

As we can see, we have two great possible horizons to continue developing technology in this field; The SECURED project ended in December 2016 but we hope that there may be opportunities to continue developing this created environment or its different improvements.

Appendix A

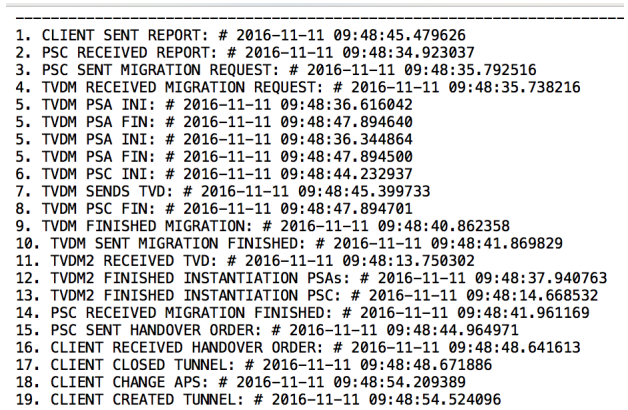
Results of the tests

```
-----------------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:37:55.079499
2. PSC RECEIVED REPORT: # 2016-11-11 09:37:44.379351
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:37:45.170838
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:37:45.337309
5. TVDM PSA INI: # 2016-11-11 09:37:46.227240
5. TVDM PSA FIN: # 2016-11-11 09:37:57.091311
5. TVDM PSA INI: # 2016-11-11 09:37:45.971840
5. TVDM PSA FIN: # 2016-11-11 09:37:57.091160
6. TVDM PSC INI: # 2016-11-11 09:37:52.896464
7. TVDM SENDS TVD: # 2016-11-11 09:37:54.020668
8. TVDM PSC FIN: # 2016-11-11 09:37:57.091391
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:37:50.368251
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:37:50.530078
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:37:23.021196
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:37:46.597784
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:37:23.888958
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:37:50.391198
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:37:54.394167
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:37:58.277512
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:37:58.423710
18. CLIENT CHANGE APS: # 2016-11-11 09:37:59.407640
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:38:04.417397
```

Figure A.1: Execution nº20 made on 11-11-2016

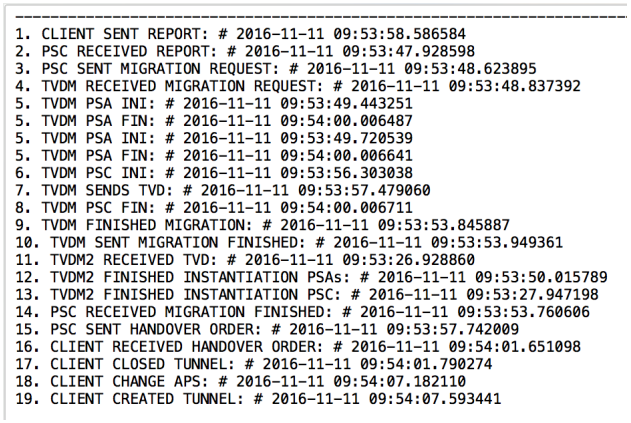
```
-----------------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:44:56.345353
2. PSC RECEIVED REPORT: # 2016-11-11 09:44:45.900477
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:44:46.772285
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:44:46.568542
5. TVDM PSA INI: # 2016-11-11 09:44:47.114114
5. TVDM PSA FIN: # 2016-11-11 09:44:57.921270
5. TVDM PSA INI: # 2016-11-11 09:44:47.371913
5. TVDM PSA FIN: # 2016-11-11 09:44:57.921414
6. TVDM PSC INI: # 2016-11-11 09:44:54.160227
7. TVDM SENDS TVD: # 2016-11-11 09:44:55.335334
8. TVDM PSC FIN: # 2016-11-11 09:44:57.921476
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:44:51.391872
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:44:51.784851
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:44:24.568939
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:44:47.893789
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:44:25.477473
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:44:52.021089
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:44:56.039809
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:44:59.545952
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:44:59.710910
18. CLIENT CHANGE APS: # 2016-11-11 09:45:01.129429
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:45:01.436069
```

Figure A.2: Execution nº21 made on 11-11-2016



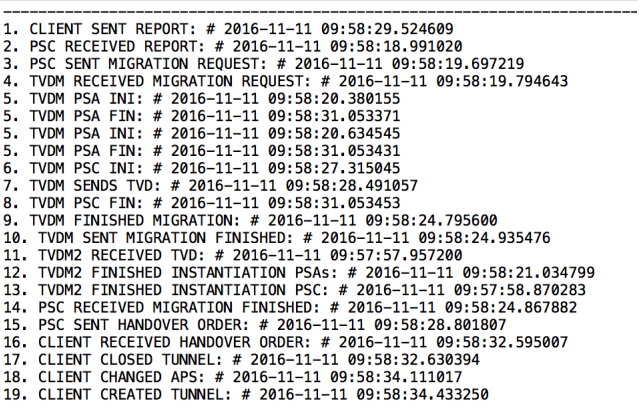
```
1. CLIENT SENT REPORT: # 2016-11-11 09:48:45.479626
2. PSC RECEIVED REPORT: # 2016-11-11 09:48:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:48:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:48:35.738216
5. TVDM PSA INI: # 2016-11-11 09:48:36.616042
5. TVDM PSA FIN: # 2016-11-11 09:48:47.894640
5. TVDM PSA INI: # 2016-11-11 09:48:36.344864
5. TVDM PSA FIN: # 2016-11-11 09:48:47.894500
6. TVDM PSC INI: # 2016-11-11 09:48:44.232937
7. TVDM SENDS TVD: # 2016-11-11 09:48:45.399733
8. TVDM PSC FIN: # 2016-11-11 09:48:47.894701
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:48:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:48:41.869829
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:48:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:48:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:48:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:48:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:48:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:48:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:48:48.671886
18. CLIENT CHANGE APS: # 2016-11-11 09:48:54.209389
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:48:54.524096
```

Figure A.3: Execution nº22 made on 11-11-2016



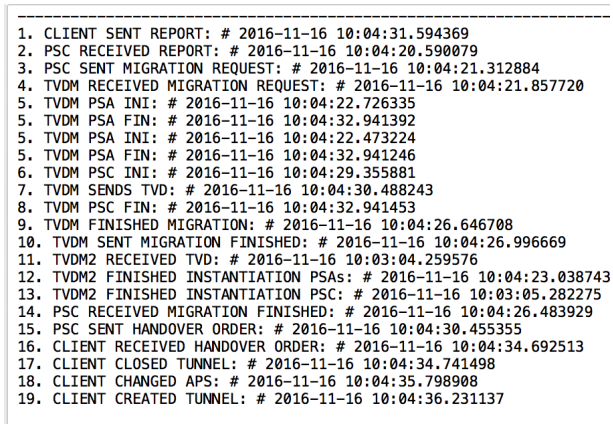
```
1. CLIENT SENT REPORT: # 2016-11-11 09:53:58.586584
2. PSC RECEIVED REPORT: # 2016-11-11 09:53:47.928598
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:53:48.623895
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:53:48.837392
5. TVDM PSA INI: # 2016-11-11 09:53:49.443251
5. TVDM PSA FIN: # 2016-11-11 09:54:00.006487
5. TVDM PSA INI: # 2016-11-11 09:53:49.720539
5. TVDM PSA FIN: # 2016-11-11 09:54:00.006641
6. TVDM PSC INI: # 2016-11-11 09:53:56.303038
7. TVDM SENDS TVD: # 2016-11-11 09:53:57.479060
8. TVDM PSC FIN: # 2016-11-11 09:54:00.006711
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:53:53.845887
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:53:53.949361
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:53:26.928860
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:53:50.015789
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:53:27.947198
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:53:53.760606
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:53:57.742009
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:54:01.651098
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:54:01.790274
18. CLIENT CHANGE APS: # 2016-11-11 09:54:07.182110
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:54:07.593441
```

Figure A.4: Execution nº23 made on 11-11-2016



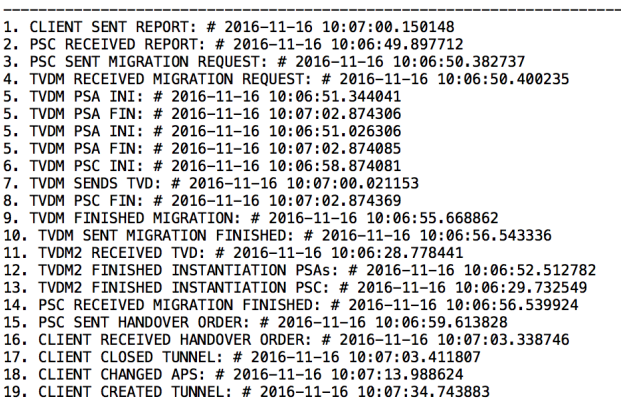
```
1. CLIENT SENT REPORT: # 2016-11-11 09:58:29.524609
2. PSC RECEIVED REPORT: # 2016-11-11 09:58:18.991020
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:58:19.697219
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:58:19.794643
5. TVDM PSA INI: # 2016-11-11 09:58:20.380155
5. TVDM PSA FIN: # 2016-11-11 09:58:31.053371
5. TVDM PSA INI: # 2016-11-11 09:58:20.634545
5. TVDM PSA FIN: # 2016-11-11 09:58:31.053431
6. TVDM PSC INI: # 2016-11-11 09:58:27.315045
7. TVDM SENDS TVD: # 2016-11-11 09:58:28.491057
8. TVDM PSC FIN: # 2016-11-11 09:58:31.053453
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:58:24.795600
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:58:24.935476
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:57:57.957200
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:58:21.034799
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:57:58.870283
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:58:24.867882
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:58:28.801807
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:58:32.595007
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:58:32.630394
18. CLIENT CHANGED APS: # 2016-11-11 09:58:34.111017
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:58:34.433250
```

Figure A.5: Execution nº24 made on 11-11-2016



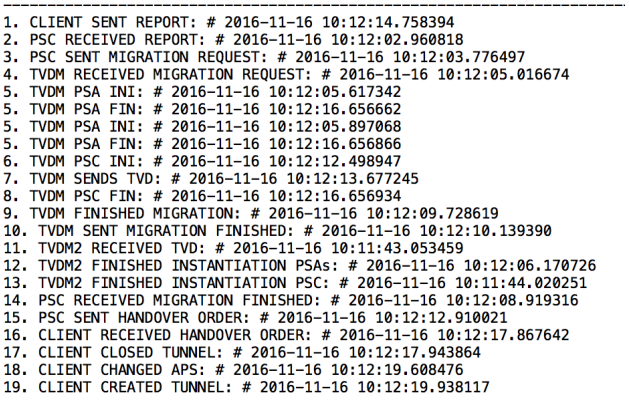
```
1. CLIENT SENT REPORT: # 2016-11-16 10:04:31.594369
2. PSC RECEIVED REPORT: # 2016-11-16 10:04:20.590079
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:04:21.312884
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:04:21.857720
5. TVDM PSA INI: # 2016-11-16 10:04:22.726335
5. TVDM PSA FIN: # 2016-11-16 10:04:32.941392
5. TVDM PSA INI: # 2016-11-16 10:04:22.473224
5. TVDM PSA FIN: # 2016-11-16 10:04:32.941246
6. TVDM PSC INI: # 2016-11-16 10:04:29.355881
7. TVDM SENDS TVD: # 2016-11-16 10:04:30.488243
8. TVDM PSC FIN: # 2016-11-16 10:04:32.941453
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:04:26.646708
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:04:26.996669
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:03:04.259576
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:04:23.038743
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:03:05.282275
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:04:26.483929
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:04:30.455355
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:04:34.692513
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:04:34.741498
18. CLIENT CHANGED APS: # 2016-11-16 10:04:35.798908
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:04:36.231137
```

Figure A.6: Execution nº44 made on 16-11-2016



```
1. CLIENT SENT REPORT: # 2016-11-16 10:07:00.150148
2. PSC RECEIVED REPORT: # 2016-11-16 10:06:49.897712
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:06:50.382737
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:06:50.400235
5. TVDM PSA INI: # 2016-11-16 10:06:51.344041
5. TVDM PSA FIN: # 2016-11-16 10:07:02.874306
5. TVDM PSA INI: # 2016-11-16 10:06:51.026306
5. TVDM PSA FIN: # 2016-11-16 10:07:02.874085
6. TVDM PSC INI: # 2016-11-16 10:06:58.874081
7. TVDM SENDS TVD: # 2016-11-16 10:07:00.021153
8. TVDM PSC FIN: # 2016-11-16 10:07:02.874369
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:06:55.668862
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:06:56.543336
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:06:28.778441
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:06:52.512782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:06:29.732549
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:06:56.539924
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:06:59.613828
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:07:03.338746
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:07:03.411807
18. CLIENT CHANGED APS: # 2016-11-16 10:07:13.988624
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:07:34.743883
```

Figure A.7: Execution nº45 made on 16-11-2016



```
1. CLIENT SENT REPORT: # 2016-11-16 10:12:14.758394
2. PSC RECEIVED REPORT: # 2016-11-16 10:12:02.960818
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:12:03.776497
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:12:05.016674
5. TVDM PSA INI: # 2016-11-16 10:12:05.617342
5. TVDM PSA FIN: # 2016-11-16 10:12:16.656662
5. TVDM PSA INI: # 2016-11-16 10:12:05.897068
5. TVDM PSA FIN: # 2016-11-16 10:12:16.656866
6. TVDM PSC INI: # 2016-11-16 10:12:12.498947
7. TVDM SENDS TVD: # 2016-11-16 10:12:13.677245
8. TVDM PSC FIN: # 2016-11-16 10:12:16.656934
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:12:09.728619
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:12:10.139390
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:11:43.053459
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:12:06.170726
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:11:44.020251
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:12:08.919316
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:12:12.910021
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:12:17.867642
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:12:17.943864
18. CLIENT CHANGED APS: # 2016-11-16 10:12:19.608476
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:12:19.938117
```

Figure A.8: Execution nº46 made on 16-11-2016

```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-16 10:57:05.531005
2. PSC RECEIVED REPORT: # 2016-11-16 10:56:54.861177
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:56:55.520417
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:56:55.783313
5. TVDM PSA INI: # 2016-11-16 10:56:56.411813
5. TVDM PSA FIN: # 2016-11-16 10:57:08.069434
5. TVDM PSA INI: # 2016-11-16 10:56:56.689222
5. TVDM PSA FIN: # 2016-11-16 10:57:08.069542
6. TVDM PSC INI: # 2016-11-16 10:57:04.370320
7. TVDM SENDS TVD: # 2016-11-16 10:57:05.538566
8. TVDM PSC FIN: # 2016-11-16 10:57:08.069582
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:57:00.960346
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:57:01.974575
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:56:33.975503
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:56:57.971758
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:56:34.948030
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:57:01.732680
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:57:04.616133
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:57:08.595589
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:57:08.638575
18. CLIENT CHANGED APS: # 2016-11-16 10:57:14.231371
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:57:14.775897
```

Figure A.9: Execution nº47 made on 16-11-2016

```
├-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-16 13:50:13.645237
2. PSC RECEIVED REPORT: # 2016-11-16 13:50:02.932151
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 13:50:03.639777
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 13:50:03.863320
5. TVDM PSA INI: # 2016-11-16 13:50:04.526736
5. TVDM PSA FIN: # 2016-11-16 13:50:15.496560
5. TVDM PSA INI: # 2016-11-16 13:50:04.778907
5. TVDM PSA FIN: # 2016-11-16 13:50:15.496688
6. TVDM PSC INI: # 2016-11-16 13:50:11.386086
7. TVDM SENDS TVD: # 2016-11-16 13:50:12.577279
8. TVDM PSC FIN: # 2016-11-16 13:50:15.496738
9. TVDM FINISHED MIGRATION: # 2016-11-16 13:50:08.938921
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 13:50:09.033690
11. TVDM2 RECEIVED TVD: # 2016-11-16 13:49:41.996291
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 13:50:04.787727
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 13:49:42.967178
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 13:50:08.851430
15. PSC SENT HANDOVER ORDER: # 2016-11-16 13:50:12.871162
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 13:50:16.824559
17. CLIENT CLOSED TUNNEL: # 2016-11-16 13:50:16.868726
18. CLIENT CHANGED APS: # 2016-11-16 13:50:22.432739
19. CLIENT CREATED TUNNEL: # 2016-11-16 13:50:22.988181
```

Figure A.10: Execution nº48 made on 16-11-2016

```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-12-01 13:40:59.603435
2. PSC RECEIVED REPORT: # 2016-12-01 13:40:49.191078
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 13:40:49.977448
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 13:40:49.833042
5. TVDM PSA INI: # 2016-12-01 13:40:50.746877
5. TVDM PSA FIN: # 2016-12-01 13:41:00.932879
5. TVDM PSA INI: # 2016-12-01 13:40:50.459558
5. TVDM PSA FIN: # 2016-12-01 13:41:00.932739
6. TVDM PSC INI: # 2016-12-01 13:40:57.287511
7. TVDM SENDS TVD: # 2016-12-01 13:40:58.463298
8. TVDM PSC FIN: # 2016-12-01 13:41:00.932945
9. TVDM FINISHED MIGRATION: # 2016-12-01 13:40:54.428560
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 13:40:54.947463
11. TVDM2 RECEIVED TVD: # 2016-12-01 13:40:17.662428
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 13:40:50.734782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 13:40:18.569856
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 13:40:55.107131
15. PSC SENT HANDOVER ORDER: # 2016-12-01 13:40:59.126046
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 13:41:02.724006
17. CLIENT CLOSED TUNNEL: # 2016-12-01 13:41:02.761684
18. CLIENT CHANGED APS: # 2016-12-01 13:41:04.227252
19. CLIENT CREATED TUNNEL: # 2016-12-01 13:41:04.706947
```

Figure A.11: Execution nº35 made on 01-12-2016

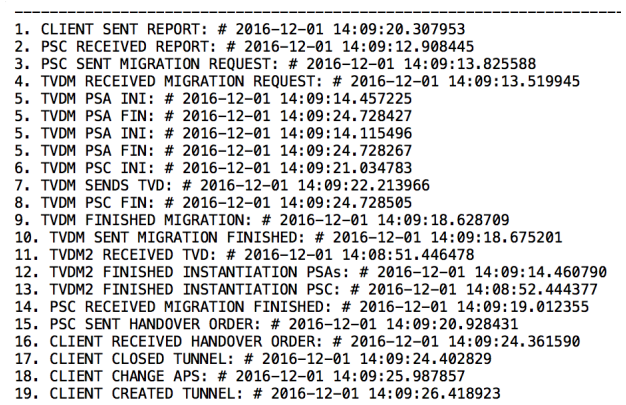


Figure A.12: Execution nº36 made on 01-12-2016

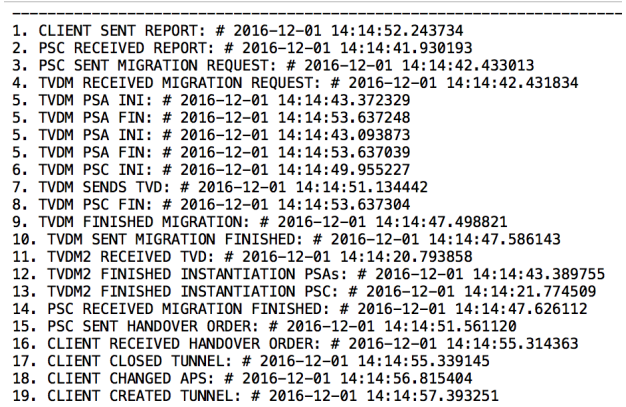


Figure A.13: Execution nº37 made on 01-12-2016

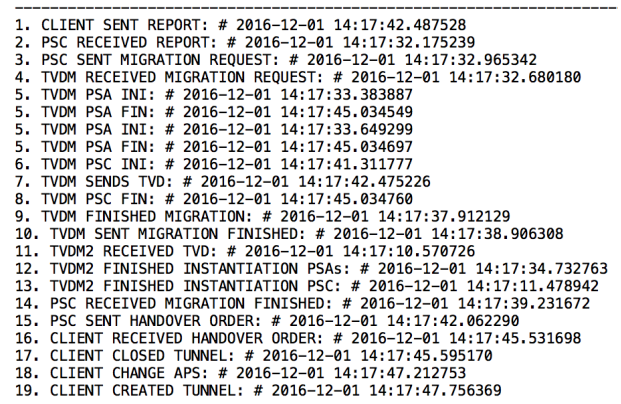


Figure A.14: Execution nº38 made on 16-11-2016

```
1. CLIENT SENT REPORT: # 2016-12-01 14:48:45.479626
2. PSC RECEIVED REPORT: # 2016-12-01 14:48:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 14:48:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 14:48:35.738216
5. TVDM PSA INI: # 2016-12-01 14:48:36.616042
5. TVDM PSA FIN: # 2016-12-01 14:48:47.894640
5. TVDM PSA INI: # 2016-12-01 14:48:36.344864
5. TVDM PSA FIN: # 2016-12-01 14:48:47.894500
6. TVDM PSC INI: # 2016-12-01 14:48:44.232937
7. TVDM SENDS TVD: # 2016-12-01 14:48:45.399733
8. TVDM PSC FIN: # 2016-12-01 14:48:47.894701
9. TVDM FINISHED MIGRATION: # 2016-12-01 14:48:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 14:48:41.869829
11. TVDM2 RECEIVED TVD: # 2016-12-01 14:48:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 14:48:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 14:48:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 14:48:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-12-01 14:48:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 14:48:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-12-01 14:48:48.671886
18. CLIENT CHANGE APS: # 2016-12-01 14:48:54.209389
19. CLIENT CREATED TUNNEL: # 2016-12-01 14:48:54.524096
```

Figure A.15: Execution nº39 made on 01-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:57:05.531005
2. PSC RECEIVED REPORT: # 2016-12-14 11:56:54.861177
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:56:55.520417
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:56:55.783313
5. TVDM PSA INI: # 2016-12-14 11:56:56.411813
5. TVDM PSA FIN: # 2016-12-14 11:57:08.069434
5. TVDM PSA INI: # 2016-12-14 11:56:56.689222
5. TVDM PSA FIN: # 2016-12-14 11:57:08.069542
6. TVDM PSC INI: # 2016-12-14 11:57:04.370320
7. TVDM SENDS TVD: # 2016-12-14 11:57:05.538566
8. TVDM PSC FIN: # 2016-12-14 11:57:08.069582
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:57:00.960346
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:57:01.974575
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:56:33.975503
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:56:57.971758
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:56:34.948030
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:57:01.732680
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:57:04.616133
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:57:08.595589
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:57:08.638575
18. CLIENT CHANGE APS: # 2016-12-14 11:57:14.231371
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:57:14.775897
```

Figure A.16: Execution nº26made on 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:15:00.150148
2. PSC RECEIVED REPORT: # 2016-12-14 11:14:49.897712
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:14:50.382737
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:14:50.400235
5. TVDM PSA INI: # 2016-12-14 11:14:51.344041
5. TVDM PSA FIN: # 2016-12-14 11:15:02.874306
5. TVDM PSA INI: # 2016-12-14 11:14:51.026306
5. TVDM PSA FIN: # 2016-12-14 11:15:02.874085
6. TVDM PSC INI: # 2016-12-14 11:14:58.874081
7. TVDM SENDS TVD: # 2016-12-14 11:15:00.021153
8. TVDM PSC FIN: # 2016-12-14 11:15:02.874369
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:14:55.668862
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:14:56.543336
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:14:28.778441
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:14:52.512782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:14:29.732549
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:14:56.539924
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:14:59.613828
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:15:03.338746
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:15:03.411807
18. CLIENT CHANGE APS: # 2016-12-14 11:15:13.988624
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:15:34.743883
```

Figure A.17: Execution nº27 made on 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:22:59.603435
2. PSC RECEIVED REPORT: # 2016-12-14 11:22:49.191078
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:22:49.977448
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:22:49.833042
5. TVDM PSA INI: # 2016-12-14 11:22:50.746877
5. TVDM PSA FIN: # 2016-12-14 11:23:00.932879
5. TVDM PSA INI: # 2016-12-14 11:22:50.459558
5. TVDM PSA FIN: # 2016-12-14 11:23:00.932739
6. TVDM PSC INI: # 2016-12-14 11:22:57.287511
7. TVDM SENDS TVD: # 2016-12-14 11:22:58.463298
8. TVDM PSC FIN: # 2016-12-14 11:23:00.932945
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:22:54.428560
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:22:54.947463
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:22:17.662428
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:22:50.734782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:22:18.569856
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:22:55.107131
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:22:59.126046
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:23:02.724006
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:23:02.761684
18. CLIENT CHANGE APS: # 2016-12-14 11:23:04.227252
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:23:04.706947
```

Figure A.18: Execution nº28 made on 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:33:55.079499
2. PSC RECEIVED REPORT: # 2016-12-14 11:33:44.379351
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:33:45.170838
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:33:45.337309
5. TVDM PSA INI: # 2016-12-14 11:33:46.227240
5. TVDM PSA FIN: # 2016-12-14 11:33:57.091311
5. TVDM PSA INI: # 2016-12-14 11:33:45.971840
5. TVDM PSA FIN: # 2016-12-14 11:33:57.091160
6. TVDM PSC INI: # 2016-12-14 11:33:52.896464
7. TVDM SENDS TVD: # 2016-12-14 11:33:54.020668
8. TVDM PSC FIN: # 2016-12-14 11:33:57.091391
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:33:50.368251
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:33:50.530078
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:33:23.021196
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:33:46.597784
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:33:23.888958
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:33:50.391198
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:33:54.394167
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:33:58.277512
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:33:58.423710
18. CLIENT CHANGE APS: # 2016-12-14 11:33:59.407640
19. CLIENT CREATED TUNNEL: # 2016-12-14 09:38:04.417397
```

Figure A.19: Execution nº29 made on 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:55:45.479626
2. PSC RECEIVED REPORT: # 2016-12-14 11:55:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:55:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:55:35.738216
5. TVDM PSA INI: # 2016-12-14 11:55:36.616042
5. TVDM PSA FIN: # 2016-12-14 11:55:47.894640
5. TVDM PSA INI: # 2016-12-14 11:55:36.344864
5. TVDM PSA FIN: # 2016-12-14 11:55:47.894500
6. TVDM PSC INI: # 2016-12-14 11:55:44.232937
7. TVDM SENDS TVD: # 2016-12-14 11:55:45.399733
8. TVDM PSC FIN: # 2016-12-14 11:55:47.894701
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:55:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:55:41.869829
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:55:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:55:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:55:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:55:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:55:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:55:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:55:48.671886
18. CLIENT CHANGE APS: # 2016-12-14 11:55:54.209389
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:55:54.524096
```

Figure A.20: Execution nº48 made on 14-12-2016

Appendix B

Project codes

B.1 Codes with added functionalities

Code snippet B.1: mainIPSEC.py

```
'''
Created on 23/mag/2014

@author: rbonafiglia

Main script to launch from gunicorn to star the orchestrator WSGI server
'''
import falcon
import Config
from Orchestrator import Orchestrator
from GraphInstatiator import GraphInstantiator
from PSAcreation import PSAcreation
from GraphInfo import GraphInfo
from PSAConf import PSAConf
###migration code###
from TVDMigration import TVDMigration
###
from VerifierCache import VerifierCache
import logging
import datetime as dt
import signal
import sys

# TODO: control if there are sessions open, if true close them
class MyFormatter(logging.Formatter):
    converter=dt.datetime.fromtimestamp
    def formatTime(self, record, datefmt=None):
        ct = self.converter(record.created)
        if datefmt:
            s = ct.strftime(datefmt)
        else:
            t = ct.strftime("%Y-%m-%d %H:%M:%S")
            s = "%s,%03d" % (t, record.msecs)
        return s


```

```
conf = Config.Configuration()
#logging.config.fileConfig(conf.LOG_FILE)
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)

fh = logging.FileHandler(conf.LOG_FILE)
fh.setLevel(logging.DEBUG)

console = logging.StreamHandler()

formatter = MyFormatter(fmt='%(asctime)s %(message)s',datefmt='%Y-%m-%d,%H:%M:%S.%f')
fh.setFormatter(formatter)
console.setFormatter(formatter)

logger.addHandler(console)
logger.addHandler(fh)
#logging.basicConfig(filename=conf.LOG_FILE,level=logging.DEBUG, format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')
logger.info("--------")
logger.info("NED / TVDM init.")
logger.info("NED / TVDM VERSION: " + str(conf.TVDM_VERSION))
logger.info("--------")
# Falcon starts

app = falcon.API()
instantiator = GraphInstantiator(conf, logger, True)
###start migration code###
migration = TVDMigration(instantiator, conf, logger)
###end migration code###
def signal_term_handler(signal, frame):
    logger.info("SIG TERM")
    if instantiator.signal_term_handler():
        logger.info("FINISH DESTROY ALL TVD")
            sys.exit(0)

signal.signal(signal.SIGTERM, signal_term_handler)
###migration code -> added parameter migration###
orch = Orchestrator(instantiator, migration)
###

### Verifier cache
vc = VerifierCache(conf)
vc.start()
app.add_route('/verify', vc)

app.add_route('/instantiateTVD', orch)
###migration code -> added new call###
app.add_route('/migration', orch)
###
psa = PSAcreation(instantiator)
app.add_route('/createPSA', psa)

graph = GraphInfo(instantiator,conf.USER_GRAPH_LOCATION, conf)
app.add_route('/getGraph', graph)
```

```
90
91 psaConfRes = PSAConf(conf.PSA_CONF_LOCATION, logger, conf,
       instantiator)
92 app.add_route('/getConf/{psa_id}/{conf_id}', psaConfRes)
```

Code snippet B.2: Orchstrator.py

```
1  import falcon
2  import json
3  from threading import Thread
4  from threading import Event
5  import thread
6  from GraphInstatiator import GraphInstantiator
7  import logging
8  import sys
9  import subprocess
10 import time
11 from datetime import datetime
12
13
14 class Orchestrator(object):
15     '''
16     Orchestrator class that intercept the REST call through the
           WSGI server
17     '''
18
19     def __init__(self, instantiator, TVDMigration):
20         '''
21         Constructor
22         '''
23         self.instantiator = instantiator
24         self.TVDMigration = TVDMigration
25         self.migrate = False
26         self.eventList = {}
27         self.handoverEvents = {}
28         self.obj = {}
29         self.obj2 = {}
30
31     def on_delete(self, request, response):
32         try:
33             args = request.stream.read()
34             self.instantiator.logger.info(request.method + " " +
                   request.uri + " " + args)
35             session = json.loads(args, 'utf-8')
36             token = self.instantiator.IPandUser[session["IP"]]
37             newTVD = Thread(target=self.instantiator.deleteUser,
                   kwargs={"session": session})
38             newTVD.start()
39             response.status = falcon.HTTP_200
40         except Exception as e:
41             self.instantiator.logger.exception(sys.exc_info()[0])
42             response.status = falcon.HTTP_501
43
44     ######start migration code############
45
46     def on_put(self, request, response):
```

```
47          '''
48          shared by the instantiation and migration of TVD
49          '''
50          try:
51              self.instantiator.timestamps["tvdm2_recieves_TVD"] =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
52              args = request.stream.read()
53              session = json.loads(args, 'utf-8')
54              self.instantiator.logger.info(request.method + " " +
                    request.uri + " " + json.dumps(session, indent=4,
                    sort_keys=True))
55              if "action" in session:
56                  if session["action"] == "TVD":
57                      newTVD = Thread(target=self.instantiator.
                            instatiateTVD, kwargs={"session": session
                            })
58                      newTVD.start()
59                      response.status = falcon.HTTP_200
60                  else:
61                      self.instantiator.timestamps = session["
                            timestamps"]
62                      response.status = falcon.HTTP_200
63              else:
64                  newTVD = Thread(target=self.instantiator.
                        instatiateTVD, kwargs={"session": session})
65                  newTVD.start()
66                  response.status = falcon.HTTP_200
67          except Exception as e:
68              self.instantiator.logger.exception(sys.exc_info()[0])
69              response.status = falcon.HTTP_501
70
71      def on_post(self, request, response):
72          '''
73          exclusive for migration
74          '''
75          try:
76              self.instantiator.tvdm_receives_migration_request =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
77              args = request.stream.read()
78              self.TVDMigration.instantiator.logger.info(request.
                    method + " " + request.uri + " " + args)
79              session = json.loads(args, 'utf-8')
80
81              self.eventList[session["token"]] = Event()
82              self.handoverEvents[session["token"]] = Event()
83              migration = Thread(name=session["token"], target=self.
                    TVDMigration.init_migration,
84                                 args=(self.eventList[session["token
                                        "]], self.handoverEvents[
                                        session["token"]],), kwargs={"
                                        session": session})
85              migration.start()
86              response.status = falcon.HTTP_200
87          except:
88              self.TVDMigration.instantiator.logger.exception(sys.
                    exc_info()[0])
```

```
89              response.status = falcon.HTTP_500
90
91      def get_client_address(self, environ):
92
93          try:
94
95              return environ['HTTP_X_FORWARDED_FOR'].split(',')[-1].
                    strip()
96
97          except KeyError:
98
99              return environ['REMOTE_ADDR']
100
101
102
103     def on_get(self, request, response):
104         '''
105         Exclusive for migration, non-bloking query migration
106         '''
107         try:
108             self.TVDMigration.instantiator.logger.info(request.
                    method + " " + request.uri)
109             user = request.get_param("user")
110             action = request.get_param("action")
111
112             obj = {}
113             if action == "migration":
114                 if user in self.eventList.keys():
115                     self.TVDMigration.instantiator.logger.info("
                            user: %s, eventList: %s" % (str(user), str
                            (self.eventList[user].isSet())))
116                     obj = {"status": self.eventList[user].isSet()}
117                     if self.eventList[user].isSet() is True:
118                         self.instantiator.
                                tvdm_sends_migration_finished =
                                datetime.utcnow().strftime("%Y-%m-%d %
                                H:%M:%S.%f")
119                         self.handoverEvents[user].set()
120                     else:
121                         obj = {"status": False}
122             else:
123                 obj["timestamps"] = self.instantiator.timestamps
124             response.body = json.dumps(obj)
125
126             self.instantiator.logger.info(str(response.body))
127             response.status = falcon.HTTP_200
128
129         except Exception as e:
130             self.instantiator.logger.exception(sys.exc_info()[0])
131             self.instantiator.logger.exception(str(e))
132             response.status = falcon.HTTP_501
133     #######end migration code###############
```

Code snippet B.3: GraphInstatiator.py

```
1 '''
```

```
Created on 29/lug/2014

'''
import json
import subprocess, shlex
from Compute import Compute
from Network import Network
from userTVD import UserTVD
from userTVDIPSECLess import UserTVD as UserTVDIPSECLess
import requests
import ast
import commands
import logging
from datetime import datetime

class GraphInstantiator(object):
    '''
    Class used to instantiate the TVD and the Profile Graph
    '''

    def __init__(self, configure, logger, useIPSEC=False):
        '''
        Constructor
        '''

        self.client_ip = None
        subprocess.call(["bash", configure.SCRIPT_LOCATION + "
            emptyDHCP.sh"])
        self.useIPSEC = useIPSEC
        self.IPandUser = {}  # Used to associate the token to the
            IP (for the logoutelf.IPandUser = {}  # Used to
            associate the token to the IP (for the logout
        self.logger = logger
        self.config = configure
        self.networkManager = Network(configure)
        self.computeManager = Compute(configure)
        self.userTVDs = {}  # All the generated TVD
        self.TokenIP = {}  # Associate the PSC IP to a specifc
            user
        self.sigTerm = False
        self.migration = False #flag migration
        self.psa_ip_route_table = int(configure.PSA_IP_ROUTE_TABLE
            )
        self.default_ssid = configure.DEFAULT_SSID
        self.local_info = configure.migration_ned_info(self.
            default_ssid)
        if type(self.local_info) is str:
            self.local_info = ast.literal_eval(self.local_info)
        self.local_host = self.local_info['ned_ip']
        self.local_port = int(self.local_info['ned_port'])
        self.defaultnamespace = configure.DEFAULT_NAME_SPACE
        self.default_ssid = configure.DEFAULT_SSID
        self.mobility = configure.MOBILITY
        self.ip_ext = configure.IP_EXT
        self.gw_ip = self.config.GATEWAY_IP
        #####test#####
```

```
52          self.tvdm_sends_migration_finished  = None
53          self.tvdm_receives_migration_request = None
54          self.psas_tt = None
55          self.tvdm_ini_psc = None
56          self.tvdm_sends_TVD = None
57          self.tvdm_fin_psc = None
58          self.tvdm_finishes_migration = None
59          self.timestamps = {}
60
61
62
63
64      def instatiateTVD(self, session):
65          '''
66          Intantiate the TVD for a logged user.
67          In case for a TVD already instatiated it will generate the
                flows used for reedirecting the traffic of the
68          new defice on it
69          '''
70          self.client_ip = str(session['IP'])
71
72
73          if 'migration' in session:
74              self.migration = bool(session['migration'])
75              self.logger.info("INSTANTIATE TVD WITH migration=%s" %
                    (str(session['migration'])))
76
77          ###start migration code###
78          if self.migration:
79              self.logger.info("migration TVD " + str(session['IP'])
                    )
80          else:
81              self.logger.info("instantiation TVD")
82          ###end migration code###
83
84          self.logger.info("IP " + session['IP'])
85          if not self.migration:
86              if session['IP'] in self.IPandUser.keys():
87                  self.logger.info("Machine already logged")
88                  return
89
90          self.IPandUser[session['IP']] = session['token']
91
92          if session['token'] in self.userTVDs.keys():
93              self.logger.info("psaLIST %s" % (str(session['PSAs']))
                    )
94
95              userTVD = self.userTVDs[session['token']]
96              if self.migration:
97                  userTVD.migration = self.migration
98
99              if userTVD.migration:
100                 self.logger.info("userTVD migration %s" % (str(
                        userTVD.migration)))
101             userTVD.addNewIP(str(session['IP']))
102             userTVD.generatePSCflows()
```

```
103             self.instantiatePSA(session)
104             return
105 
106         ###start migraton Code###
107          if self.migration:  # If the user TVD migrated
108             vlanID = session['vlanID']
109          else:
110          ###end migration code###
111             # If the user have no TVD instantiated
112             vlanID = self.networkManager.getNewVLANID()
113 
114          ###start migraton Code###
115          if self.migration:
116             userInterface = session['userInterface']
117             self.logger.info("userInterface    " + userInterface)
118          else:
119          ###end migration code###
120             userInterface = None
121 
122          if self.useIPSEC:
123             ###migration code->added parameters: userInterface and
                    self.migration###
124             newTVD = UserTVD(session['token'], vlanID, self.
                networkManager, self.computeManager, self.config,
125                              self.logger, userInterface, self.
                                 migration, self.mobility)
126 
127          else:
128             ###migration code->added parameters: userInterface and
                    self.migration###
129             newTVD = UserTVDIPSECLess(session['token'], vlanID,
                self.networkManager, self.computeManager, self.
                config,
130                                       self.logger, userInterface,
                                          self.migration)
131 
132 
133          self.userTVDs[session['token']] = newTVD
134 
135          # Generates the PSC for the user
136          ###start migration code###
137          if self.migration:
138             mac = session['psc']['interfaces'][0]['mac']
139          else:
140          ###end migration code###
141             mac = self.networkManager.generateMACaddress()
142 
143          ###start migration code###
144          if self.migration:
145             newPSC = session['psc']
146          else:
147          ###end migration code###
148             newPSC = self.definePSC(vlanID, mac)
149 
150          ###start migration code###
151          if self.migration:
```

```
152             pscAddr = session['pscAddr']
153         else:
154         ###end migration code###
155             pscAddr = self.networkManager.configureNewIPOnDHCP(mac
                    , session['token'])
156 
157         self.logger.info("PSC address for user " + session['token
                '] + " " + pscAddr)
158         self.TokenIP[pscAddr] = session['token']
159         ###start migration code###
160         if self.migration:
161             pscName = session['pscName']
162         else:
163         ###end migration code###
164             pscName = self.computeManager.instantiateNF('psc',
                    newPSC)
165 
166         ##newPSC['name'] = pscName
167         ##newTVD.setPSC(newPSC, pscAddr)
168         ##self.logger.info("PSC created")
169 
170         newPSC['name'] = pscName
171         newTVD.setPSC(newPSC, pscAddr)
172         newTVD.generatePSCflows()
173 
174         self.timestamps["tvdm2_finish_instantiation_PSC"] =
                datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
175         self.logger.info("PSC created")
176 
177         newTVD.addNewIP(str(session['IP']))
178 
179         ###start migration code###
180         if self.mobility:
181             try:
182                 headers = {'Content-Type': 'application/json', '
                        Accept': 'application/json'}
183 
184                 if not 'ssid' in session:
185                     info = self.local_info
186                     self.logger.info("NAT TAKE LOCAL_INFO: %s" %
                            str(self.local_info))
187                 else:
188                     info = self.config.migration_ned_info(session
                            ['ssid'])
189 
190                     self.logger.info("NAT TAKE SSID %s INFO: %s" %
                             (ssid,str(info)))
191 
192                 if type(info) is str:
193                         info = ast.literal_eval(info)
194                 nat = info['nat']
195                 url = "http://%s:%s/switch" % (str(nat['server
                        '][0]),str(nat['server'][1]))
196                 payload = {'route': nat['route']}
197                 self.logger.info("NAT Request URL: %s" % url)
198 
```

```
199                 self.logger.info("NAT Request Payload: %s" % str(
                        payload))
200                 r = requests.post(url, json=payload, headers=
                        headers )
201                 self.logger.info("NAT request Response: %s" % str(
                        r.status_code))
202             except Exception as e:
203                 self.logger.error("NAT request error\n%s" % (str(e
                        )))
204
205         if self.migration:
206             psaIPaddresses = session['psaIPaddresses'].items()
207             for psa in  psaIPaddresses:
208                 psaID = psa[0]
209                 ip = psa[1]
210                 newTVD.associateIPPSA(psaID, ip)
211
212
213         if self.migration:
214             self.instantiatePSA(session)
215             self.timestamps["tvdm2_finish_instantiation_PSAs"] =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
216             self.logger.info(str(self.timestamps) )
217             ###end migration code###
218
219
220
221     def definePSC(self, vlanID, mac):
222         '''
223         Define the template for the PSC of the TVD
224         '''
225
226         properties = {}
227         properties['memory'] = "512"
228         properties['vcpu'] = "1"
229         properties['interfaces'] = []
230
231         interface = {}
232         interface['mac'] = mac
233         interface['bridge'] = "brCtl"
234         interface['name'] = self.networkManager.generateVMPort()
235         properties['interfaces'].append(interface)
236
237         interface = {}
238         interface['vlan'] = vlanID
239         interface['bridge'] = "brCtl"
240         interface['name'] = self.networkManager.generateVMPort()
241         properties['interfaces'].append(interface)
242
243         interface = {}
244         interface['bridge'] = "brData"
245         interface['name'] = self.networkManager.generateVMPort()
246         properties['interfaces'].append(interface)
247
248         return properties
249
```

```
250     def deleteUser(self, session, migration=False):
251         '''
252         This function is used for the user log out. If there are
                multiple devices connected it will be deleted only the
253         flows for the specifc device that did the log out.
254         If the last device for that specific user is disconnected
                the graph will be destroyed
255         '''
256         if session['IP'] not in self.IPandUser.keys():
257             self.logger.info("Machine not logged in")
258             return
259         token = self.IPandUser[session['IP']]
260 
261         if self.mobility:
262             self.mobility_iprules(session, "delete", token)
263 
264         del self.IPandUser[session['IP']]
265         userTVD = self.userTVDs[token]
266 
267 
268         if userTVD.deleteTVD(session['IP'], migration=migration):
269             del self.TokenIP[userTVD.pscAddr]
270             del self.userTVDs[token]
271 
272     def instantiatePSA(self, session):
273         '''
274         Instantiate the PSA of the TVD
275         '''
276         global err, err
277         PSAList = session['PSAs']
278         self.logger.info("------> PSAList %s \n" %(json.dumps(
                PSAList)))
279         oup = "egress flow: "
280         if 'egress_flow' in session:
281             for psa in session['egress_flow']:
282                 oup = oup + str(psa) + ", "
283         oup = oup + "\n ingress flow: "
284         if 'ingress_flow' in session:
285             for psa in session['ingress_flow']:
286                 oup = oup + str(psa) + ", "
287         self.logger.info("\n%s\n" %(str(oup)))
288 
289         userTVD = self.userTVDs[session['token']]
290         userTVD.instantiatePSA(PSAList)
291 
292         self.logger.info("\n\n--> of FLOWS\n %s \n" %(json.dumps(
                userTVD.generatedFlows)))
293         self.logger.info("\n\n--> ofPorts\n %s \n" %(json.dumps(
                userTVD.ofPorts)))
294 
295         #######start migration code########
296         if self.mobility:
297             self.mobility_iprules(session, "add", None)
298         #######stop migration code#########
299 
300 
```

```
301     def signal_term_handler(self):
302         '''
303         Used on SIGTERM signal to destroy all the instantiated TVD
304         '''
305         if not self.sigTerm:
306             self.sigTerm = True
307             for userTVD in self.userTVDs.values():
308                 userTVD.deleteAllTVD()
309             return True
310         return False
311 
312     def get_default_gw(self, netns="default"):
313         '''
314         function to return the default gw ip
315         '''
316         strs = subprocess.check_output(shlex.split('ip netns exec
                '+ str(netns)+' ip r l'))
317         gateway = strs.split('default via')[-1].split()[0]
318         return gateway
319 
320     def mobility_iprules(self,session, adddel, token):
321 
322         if (adddel == "add"):
323             userTVD = self.userTVDs[session['token']]
324         else:
325             userTVD = self.userTVDs[token]
326 
327         psaIPaddresses = userTVD.psaIPaddresses.items()
328         ssid = self.config.DEFAULT_SSID
329         info = self.config.migration_ned_info(ssid)
330         if type(info) is str:
331             info = ast.literal_eval(info)
332         nat = info['nat']
333         try:
334             if 'default_gw' in nat:
335                 self.gw_ip = str(nat['default_gw'])
336             else:
337                 self.gw_ip = str(nat['server'][0])
338         except Exception as err:
339             self.logger.info("----> error getting the default gw "
                    + str(err))
340             self.gw_ip = str(nat['server'][0])
341 
342         for psa in psaIPaddresses:
343             psaID = psa[0]
344             ip_psa = psa[1]
345             self.logger.info("psaID " + psaID + " ip " + ip_psa)
346             if (adddel == "add"):
347                 self.psa_ip_route_table += 1
348                 if session['token'] not in userTVD.iprules or type
                    (userTVD.iprules[session['token']]) is not
                    dict:
349                     userTVD.iprules[session['token']] = {}
350                 if not ip_psa in userTVD.iprules[session['token
                    ']]:
351                     userTVD.iprules[session['token']][ip_psa] = []
```

```
352                 self.logger.info("[Mobility] ip_psa %s tables %s"
                        % (str(ip_psa), str(userTVD.iprules[session['
                        token']][ip_psa])))
353                 if not self.psa_ip_route_table in userTVD.iprules[
                        session['token']][ip_psa]:
354                     self.logger.info("Cleaning ip route table %s
                            ip_psa %s" % (str(self.psa_ip_route_table)
                            ,str(ip_psa)))
355                     commands.cleanRouteTable(table=str(self.
                            psa_ip_route_table), netNs="default")
356                 userTVD.iprules[session['token']][ip_psa].append(
                        self.psa_ip_route_table)
357                 ##add ip rule from
358                 commands.addIPrule(table=self.psa_ip_route_table,
                        addressFrom=ip_psa, pref=self.
                        psa_ip_route_table, netns="default")
359                 ##add ip rule to
360                 commands.addIPrule(table=self.psa_ip_route_table,
                        addressTo=ip_psa, pref=self.psa_ip_route_table
                        , netns="default")
361                 ##add ip routes to the table
362                 result = commands.addRoute(table=self.
                        psa_ip_route_table, addr=str(self.gw_ip),
                        default=True, netNs="default")
363                 if result is not None: self.logger.warning("\n\n[
                        mobility ip route] error add default route via
                         %s table %s" % (str(self.gw_ip), str(self.
                        psa_ip_route_table)))
364                 result = commands.addRoute(table=self.
                        psa_ip_route_table, addr=str(ip_psa), via=str(
                        self.ip_ext), netNs="default")
365                 if result is not None: self.logger.warning("\n\n[
                        mobility ip route] error add route to %s via %
                        s" % (str(self.ip_ext), str(ip_psa)))
366             else:
367                 tables = []
368                 if ip_psa in userTVD.iprules[token]:
369                     tables = userTVD.iprules[token][ip_psa]
370                     self.logger.info("Cleaning ip route table and
                            rules ip_psa %s tables %s" % (str(ip_psa),
                             str(tables)))
371                 for table in tables:
372                     commands.delIPrule(table=table, addressFrom=
                            ip_psa, netns="default")
373                     commands.delIPrule(table=table, addressTo=
                            ip_psa, netns="default")
374                     commands.cleanRouteTable(table=str(table),
                            netNs="default")
```

Code snippet B.4: userTVD.py

```
1 import commands
2 import json
3
4 from manifestManager import ManifestManager
5
```

```
6
7  class UserTVD(object):
8      '''
9      User TVD class in case of IPSEC NED configuration
10     '''
11
12     def __init__(self, userName, vlanID, networkManager,
           computeManager, config, logger, userInterface,
13                  migration=False, mobility=False):
14
15         self.logger = logger
16         self.networkManager = networkManager
17         ###start migration code###
18         if migration:
19             self.userInterface = userInterface
20             commands.createNewPort('brData', self.userInterface)
21             self.logger.info("entra migracio")
22         else:
23             ###end migration code###
24             self.userInterface = self.networkManager.generatePort
                   ('brData')
25             self.logger.info("entra intanciacio")
26
27         self.interfaceIP, result = self.networkManager.
               generateRoutingTable(self.userInterface)
28         self.logger.debug("\n\n->>> [RoutingTable] Interface %s,
               IP %s result: \n %s" % (str(self.userInterface), str(
               self.interfaceIP), str(result)))
29         self.userName = userName
30         self.userIP = []
31         self.vlanID = vlanID
32         self.pscAddr = None
33         self.psc = None
34         self.generatedFlows = []
35         self.computeManager = computeManager
36         self.config = config
37         self.psaList = []
38         self.psaIPaddresses = {}
39         self.manifestManager = ManifestManager(config)
40         self.migration = migration
41         self.iprules = {}
42         self.psaID_first = None
43         self.psaID_last = None
44         self.mobility = mobility
45         self.ofPorts = {}
46
47     def addNewIP(self, newIP):
48         '''
49             Add a new machine on the TVD with the given IP
50         '''
51         if newIP not in self.userIP:
52             self.userIP.append(newIP)
53             commands.addIPrule(table=self.userInterface,
                   addressFrom=newIP + "/32", pref=2, netns="default
                   ")
54             commands.addIPrule(table=self.userInterface, addressTo
```

```
                =newIP + "/32", pref=2, netns="default")
55
56      def delUserIP(self, newIP):
57          '''
58          Remove the flow for the given IP
59          '''
60          self.userIP.remove(newIP)
61          commands.delIPrule(table=self.userInterface, addressFrom=
                newIP + "/32", pref=2, netns="default")
62          commands.delIPrule(table=self.userInterface, addressTo=
                newIP + "/32", pref=2, netns="default")
63
64      def setPSC(self, newPSC, pscAddr):
65          '''
66      Configure the PSC of the TVD
67      '''
68          self.psc = newPSC
69          self.pscAddr = pscAddr
70          self.logger.info("User " + self.userName + " PSC addr: " +
                pscAddr)
71
72      def generatePSCflows(self):
73          '''
74          Generete the flow for the PSC
75          '''
76          flow = {}
77          bridgeName = 'brData'
78          flow['priority'] = "10"
79          match = {}
80          match['in_port'] = self.userInterface
81          self.ofPorts[self.userInterface] = commands.findPort(
                bridgeName, self.userInterface)
82          match['dl_type'] = "0x0806"
83          match['nw_dst'] = self.config.PSC_DP_IF_IP
84          flow['match'] = match
85          action = {}
86          action['output'] = self.psc['interfaces'][2]['name']
87          self.ofPorts[self.psc['interfaces'][2]['name']] = commands
                .findPort(bridgeName, self.psc['interfaces'][2]['name
                '])
88          flow['action'] = action
89          self.networkManager.generateFlow('brData', flow)
90          self.generatedFlows.append(flow)
91
92          flow = {}
93          flow['priority'] = "10"
94          match = {}
95          match['in_port'] = self.userInterface
96          match['dl_type'] = "0x0800"
97          match['nw_dst'] = self.config.PSC_DP_IF_IP
98          flow['match'] = match
99          action = {}
100         action['output'] = self.psc['interfaces'][2]['name']
101         flow['action'] = action
102         self.networkManager.generateFlow('brData', flow)
103         self.generatedFlows.append(flow)
```

```
104
105         flow = {}
106         match = {}
107         match['in_port'] = self.psc['interfaces'][2]['name']
108         flow['match'] = match
109         action = {}
110         action['output'] = self.userInterface
111         flow['action'] = action
112         self.networkManager.generateFlow('brData', flow)
113         self.generatedFlows.append(flow)
114
115     def deleteAllTVD(self):
116         '''
117         Delete the TVD
118         '''
119         self.logger.info("Deleting " + self.userName + " TVD")
120         for ip in self.userIP:
121             self.deleteTVD(ip)
122
123     def deleteTVD(self, IPaddr, migration=False):
124         '''
125         Remove an IP from the TVD in case of the last IP the TVD
               will be deleted
126         '''
127         self.logger.info("Removing IP " + IPaddr)
128         self.delUserIP(IPaddr)
129         if len(self.userIP) > 0:
130             return False
131         self.logger.info("Deleting flows: %s" % (str(self.
               generatedFlows)))
132         for flow in self.generatedFlows:
133             if migration is False:
134                 self.networkManager.deleteFlow('brData', flow)
135             else:
136                 self.networkManager.deleteFlow_migration('brData',
                       flow, ofPorts=self.ofPorts)
137
138
139         self.logger.info("Deleting the user Interface")
140         self.networkManager.deletePort('brData', self.
               userInterface)
141
142         ##self.logger.info("Deleting PSC %s\n" % (json.dumps(self.
               psc, indent=4, sort_keys=True)))
143         if self.psc is not None:
144             self.computeManager.deleteNF(self.psc['name'])
145
146         self.logger.info("Deleting PSA")
147         for psa in self.psaList:
148             self.computeManager.deleteNF(psa['name'])
149
150         logLine = "PSA:"
151         for ipAddr in self.psaIPaddresses.values():
152             logLine = logLine + " " + ipAddr
153         logLine = logLine + ", PSC: " + self.pscAddr + " removed"
154         self.logger.info(logLine)
```

```
155         return True
156 
157     def deleteFlows(self,IPaddr):
158 
159         for flow in self.generatedFlows:
160             self.networkManager.deleteFlow_migration('brData',
                    flow)
161             self.generatedFlows.remove(flow)
162 
163     def associateIPPSA(self, psaID, ip=None):
164         '''
165         Associate an IP on a PSA
166         '''
167 
168         ###start migration code####
169         if self.migration:
170             ipAddr = ip
171             self.psaIPaddresses[psaID] = ip
172         else:
173         ###end migration code###
174             ipAddr = self.networkManager.getPSAnewAddress()
175             self.psaIPaddresses[psaID] = ipAddr
176         self.logger.info("User " + self.userName + " PSA " + psaID
                + " addr: " + ipAddr)
177 
178 
179     def definePSA(self, psaID):
180         '''
181         Define a new PSA for the TVD
182         '''
183         manifest = self.manifestManager.getManifest(psaID)
184         properties = {}
185         properties['memory'] = manifest['memory']
186         properties['vcpu'] = manifest['vcpu']
187         properties['interfaces'] = []
188 
189         interface = {}
190         interface['bridge'] = "brData"
191         interface['name'] = self.networkManager.generateVMPort()
192         properties['interfaces'].append(interface)
193 
194         interface = {}
195         interface['bridge'] = "brData"
196         interface['name'] = self.networkManager.generateVMPort()
197         properties['interfaces'].append(interface)
198 
199         interface = {}
200         interface['vlan'] = self.vlanID
201         interface['bridge'] = "brCtl"
202         interface['name'] = self.networkManager.generateVMPort()
203         properties['interfaces'].append(interface)
204 
205         return properties
206 
207     def instantiatePSA(self, PSAList):
208         '''
```

```
209         Instantiate the PSAs for the TVD
210         '''
211         for psa in PSAList:
212             ###start migration code####
213             if self.migration:
214                 psaProperties = psa
215             else:
216                 ###end migration code###
217                 psaProperties = self.definePSA(psa['id'])
218             ###start migration code###
219             if self.migration:
220                 psaName = psa['name']
221                 for interface in psa['interfaces']:
222                     self.logger.info("psa interface " + interface
                            ['bridge'] + " " + interface['name'])
223                     #commands.createNewPort(interface['bridge'],
                            interface['name'])
224             else:
225                 ###end migration code###
226                 psaName = self.computeManager.instantiateNF(psa['
                        id'], psaProperties)
227             psaProperties['name'] = psaName
228             for interface in psaProperties['interfaces']:
229                 self.logger.info("psa interface " + interface['
                        bridge'] + " " + interface['name'])
230                 self.logger.info("lastinterface " + self.
                        userInterface)
231             self.psaList.append(psaProperties)
232
233         self.logger.info("\n\n [userTVD] PSAList:\n%s" % (json.
                dumps(self.psaList, indent=4, sort_keys=True)))
234         lastInterface = self.userInterface
235         for psa in self.psaList:
236             flow = {}
237             bridgeName = 'brData'
238             match = {}
239             match['in_port'] = lastInterface
240             self.ofPorts[lastInterface] = commands.findPort(
                    bridgeName, lastInterface)
241             flow['match'] = match
242             action = {}
243             action['output'] = psa['interfaces'][0]['name']
244             self.ofPorts[psa['interfaces'][0]['name']] = commands.
                    findPort(bridgeName, psa['interfaces'][0]['name'])
245             flow['action'] = action
246             self.networkManager.generateFlow('brData', flow)
247             self.generatedFlows.append(flow)
248             flow = {}
249             match = {}
250             match['in_port'] = psa['interfaces'][0]['name']
251             flow['match'] = match
252             action = {}
253             action['output'] = lastInterface
254             flow['action'] = action
255             self.logger.info("flows2 " + str(flow))
256             self.networkManager.generateFlow('brData', flow)
```

```
257                 self.generatedFlows.append(flow)
258                 lastInterface = psa['interfaces'][1]['name']
259             flow = {}
260             bridgeName = 'brData'
261             match = {}
262             match['in_port'] = lastInterface
263             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
264             flow['match'] = match
265             action = {}
266             action['output'] = self.config.EXIT_INTERFACE
267             self.ofPorts[self.config.EXIT_INTERFACE] = commands.
                    findPort(bridgeName, self.config.EXIT_INTERFACE)
268             flow['action'] = action
269             self.logger.info("flows3 " + str(flow))
270             self.networkManager.generateFlow('brData', flow)
271             self.generatedFlows.append(flow)
272 
273             flow = {}
274             flow['priority'] = "10"
275             bridgeName = 'brData'
276             match = {}
277             match['in_port'] = self.config.EXIT_INTERFACE
278             match['dl_type'] = "0x0806"
279             match['nw_dst'] = self.interfaceIP
280             flow['match'] = match
281             action = {}
282             action['output'] = lastInterface
283             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
284             flow['action'] = action
285             self.logger.info("flows4 " + str(flow))
286             self.networkManager.generateFlow('brData', flow)
287             self.generatedFlows.append(flow)
288 
289             flow = {}
290             flow['priority'] = "10"
291             bridgeName = 'brData'
292             match = {}
293             match['in_port'] = self.config.EXIT_INTERFACE
294             match['dl_type'] = "0x0800"
295             match['nw_dst'] = self.interfaceIP
296             flow['match'] = match
297             action = {}
298             action['output'] = lastInterface
299             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
300             flow['action'] = action
301             self.logger.info("flows5 " + str(flow))
302             self.networkManager.generateFlow('brData', flow)
303             self.generatedFlows.append(flow)
304 
305             for ipAddr in self.psaIPaddresses.values():
306                 flow = {}
307                 bridgeName = 'brData'
308                 flow['priority'] = "10"
```

```
309             match = {}
310             match['in_port'] = self.config.EXIT_INTERFACE
311             match['dl_type'] = "0x0806"
312             match['nw_dst'] = ipAddr
313             flow['match'] = match
314             action = {}
315             action['output'] = lastInterface
316             self.ofPorts[lastInterface] = commands.findPort(
                    bridgeName, lastInterface)
317             flow['action'] = action
318             self.logger.info("flows6 " + str(flow))
319             self.networkManager.generateFlow('brData', flow)
320             self.generatedFlows.append(flow)
321 
322             flow = {}
323             flow['priority'] = "10"
324             match = {}
325             match['in_port'] = self.config.EXIT_INTERFACE
326             match['dl_type'] = "0x0800"
327             match['nw_dst'] = ipAddr
328             flow['match'] = match
329             action = {}
330             action['output'] = lastInterface
331             flow['action'] = action
332             self.logger.info("flows7 " + str(flow))
333             self.networkManager.generateFlow('brData', flow)
334             self.generatedFlows.append(flow)
335 
336         for ipAddr in self.userIP:
337             flow = {}
338             flow['priority'] = "10"
339             match = {}
340             match['in_port'] = self.config.EXIT_INTERFACE
341             match['dl_type'] = "0x0806"
342             match['nw_dst'] = ipAddr
343             flow['match'] = match
344             action = {}
345             action['output'] = lastInterface
346             flow['action'] = action
347             self.logger.info("flows8 " + str(flow))
348             self.networkManager.generateFlow('brData', flow)
349             self.generatedFlows.append(flow)
350 
351             flow = {}
352             flow['priority'] = "10"
353             match = {}
354             match['in_port'] = self.config.EXIT_INTERFACE
355             match['dl_type'] = "0x0800"
356             match['nw_dst'] = ipAddr
357             flow['match'] = match
358             action = {}
359             action['output'] = lastInterface
360             flow['action'] = action
361             self.logger.info("flows9 " + str(flow))
362             self.networkManager.generateFlow('brData', flow)
363             self.generatedFlows.append(flow)
```

B.2 Code with the new functionalities of mobility

Code snippet B.5: TVDMigration.py

```
import ast
import datetime
import json
import libvirt
import requests
import socket
import subprocess
import threading
import time
from threading import Event
from threading import Thread
import paramiko

from libvirtManager import ComputeManager
class TVDMigration(object):
    '''
       Class used to migrate user's TVD and VMs
    '''

    def __init__(self, instantiator, configure, logger):
        self.user = None
        self.userTVDs = instantiator.userTVDs  # All the generated
            TVD
        self.remote_conn_string = ""
        self.local_conn_string = ""
        self.conn = ""
        self.remote_conn = ""

        self.default_ssid = configure.DEFAULT_SSID
        self.local_info = configure.migration_ned_info(self.
           default_ssid)
        if type(self.local_info) is str:
            self.local_info = ast.literal_eval(self.local_info)
        self.local_host = self.local_info['ned_ip']
        self.local_port = int(self.local_info['ned_port'])

        self.config = configure
        self.computeManager = ComputeManager(configure)
        self.__last_bytes = -1
        self.__last_time = datetime.time()
        self.logger = logger
        self.PSAList = {}
        self.TVD = {}
        self.namePSC = None
        self.instantiator = instantiator
        self.TokenIP = instantiator.TokenIP
        self.ssh = None
        self.PSC = None
        self.disk_base = None
        self.local_conn_string = "qemu:///system"
        self.conn = libvirt.open(self.local_conn_string)
        self.local_dom = None
```

```
51
52      def init_migration(self, e, he, session):
53          '''
54          Inicialitzation class and call migration functions
55          '''
56
57          self.logger.info("token " + str(session['token']))
58
59          self.instantiator.psas_tt = {}
60          self.remote_info = self.config.migration_ned_info(session
                ['ssid'])
61          if type(self.remote_info) is str:
62              self.remote_info = ast.literal_eval(self.remote_info)
63          self.remote_host = self.remote_info['ned_ip']
64          self.remote_port = int(self.remote_info['ned_port'])
65          self.logger.info("remote host %s:%s" % (self.remote_host,
66           self.remote_port))
67          self.PSAList = self.userTVDs[session['token']].psaList
68          self.user = self.userTVDs[session['token']].userName
69          self.namePSC = self.userTVDs[session['token']].psc['name']
70          self.PSC = self.instantiator.userTVDs[session['token']].
                psc
71          try:
72              self.ssh = self.createSSHClient(self.remote_host, self
                    .remote_port,
73               'root', 'secured')
74          except:
75              self.logger.info("Error to generate ssh")
76          i = Event()
77          self.TsendTVD = Thread(target=self.sendTVD, kwargs={"
                session": session})
78          Migration = Thread(target=self.migration, args=(i, he, ))
79          Migration.start()
80          i.wait()
81          e.set()
82          Migration.join()
83          self.logger.info("Migration FINISHED. Deleting remaining
                user data... ")
84          self.sendTimestamps(session)
85          self.logger.info("calling instantiator.deleteUser with
                session: \n%s" %
86          (json.dumps(session, indent=4, sort_keys=True)))
87          self.instantiator.deleteUser(session, migration=True)
88          self.logger.info("\n\nUSER DELETED... SHOULD BE CLEAN... \
                n")
89
90
91      def sendTimestamps(self, session):
92          obj = {"action":"timestamps"}
93
94          #4#
95          if self.instantiator.tvdm_receives_migration_request:
96              self.instantiator.timestamps["
                    tvdm_receives_migration_request"] =
97              self.instantiator.tvdm_receives_migration_request
98          #5#
```

```
99          if self.instantiator.psas_tt:
100             self.instantiator.timestamps["psas_tt"] =
101             self.instantiator.psas_tt
102         #6#
103         if self.instantiator.tvdm_ini_psc:
104             self.instantiator.timestamps["tvdm_ini_psc"] =
105             self.instantiator.tvdm_ini_psc
106         #7#
107         if self.instantiator.tvdm_sends_TVD:
108             self.instantiator.timestamps["tvdm_sends_TVD"] =
109             self.instantiator.tvdm_sends_TVD
110         #8#
111         if self.instantiator.tvdm_fin_psc:
112             self.instantiator.timestamps["tvdm_fin_psc"] =
113             self.instantiator.tvdm_fin_psc
114         #9#
115         if self.instantiator.tvdm_finishes_migration:
116             self.instantiator.timestamps["tvdm_finishes_migration
                    "] =
117             self.instantiator.tvdm_finishes_migration
118         #10#
119         if self.instantiator.tvdm_sends_migration_finished:
120             self.instantiator.timestamps["
                    tvdm_sends_migration_finished"] = self.
                    instantiator.tvdm_sends_migration_finished
121         #11---------------------#
122         '''
123         if self.instantiator.tvdm2_recieves_TVD:
124             self.instantiator.timestamps["tvdm2_recieves_TVD"] =
                    self.instantiator.tvdm2_recieves_TVD
125         #12#
126         if self.instantiator.tvdm2_finish_instantiation_PSAs:
127             self.instantiator.timestamps["
                    tvdm2_finish_instantiation_PSAs"] = self.
                    instantiator.tvdm2_finish_instantiation_PSAs
128         #13#
129         if self.instantiator.tvdm2_finish_instantiation_PSC:
130             self.instantiator.timestamps["
                    tvdm2_finish_instantiation_PSC"] = self.
                    instantiator.tvdm2_finish_instantiation_PSC
131         '''
132         obj["timestamps"] = self.instantiator.timestamps
133         try:
134             timestamps = json.dumps(obj)
135             self.logger.info("TIMESTAMPS: " + str(timestamps))
136         except:
137             self.logger.info("Error to created TVD json")
138
139         try:
140             cmd = "ip netns exec orchNet curl -X PUT --header
                    Accept: application/json --header Content-Type:
                    application/json -d '" + timestamps + "' http
                    ://192.168.1.1:8080/migration"
141             (results, errors) = self.execute_command(self.ssh, cmd
                    , False)
142
```

```
143             except:
144                 self.logger.info(errors)
145 
146         def sendTVD(self, session):
147             '''
148             Obtain user's TVD and migrate this
149             '''
150 
151             self.TVD['token'] = self.instantiator.userTVDs[session['
                    token']].userName
152             self.TVD['IP'] = session['IP']
153             self.TVD['userInterface'] =
154              self.instantiator.userTVDs[session['token']].
                    userInterface
155             self.instantiator.logger.info("userIsessionnterface " +
156              str(self.TVD['userInterface']))
157             self.TVD['vlanID'] = self.instantiator.userTVDs[session['
                    token']].vlanID
158             self.TVD['pscAddr'] =
159             self.instantiator.userTVDs[session['token']].pscAddr
160             self.TVD['psc'] = self.instantiator.userTVDs[session['
                    token']].psc
161             self.TVD['pscName'] =
162              self.instantiator.userTVDs[session['token']].psc['name']
163             self.TVD['generatedFlows'] =
164              self.instantiator.userTVDs[session['token']].
                    generatedFlows
165             self.TVD['PSAs'] = self.instantiator.userTVDs[session['
                    token']].psaList
166             self.TVD['psaIPaddresses'] =
167              self.instantiator.userTVDs[session['token']].
                    psaIPaddresses
168             self.TVD['migration'] = "True"
169             self.TVD['action'] = "TVD"
170 
171             try:
172                 TVD = json.dumps(self.TVD)
173                 self.logger.info("TVD: " + str(TVD))
174             except:
175                 self.logger.info("Error to created TVD json")
176 
177             try:
178                 cmd = "ip netns exec orchNet curl -X PUT --header \
179                 Accept: application/json --header Content-Type:
                        application/json -d\
180                  '" + TVD + "' http://192.168.1.1:8080/migration"
181                 (results, errors) = self.execute_command(self.ssh, cmd
                        , False)
182                 self.instantiator.tvdm_sends_TVD =
183                  datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%
                        S.%f")
184 
185             except:
186                 self.logger.info(errors)
187 
188         def migration(self, i, he):
```

```
189         '''
190         Prepare PSC and PSAs migration
191         '''
192
193         '''
194         PSAs migration
195         '''
196         tpsas = []
197         for psa in self.PSAList:
198             self.original = self.obtain_disk_base(psa['disk']['
                location'])
199             self.generate_remote_disk(self.original, psa['disk']['
                type'], psa['disk']['location'], psa['name'])
200             self.logger.info("going to migrate "+ psa['name'])
201             tpsa = TMigration(vm=psa['name'], sleep=0,
202                               local_host=self.local_host,
203                               remote_host=self.remote_host,
204                               remote_port=self.remote_port,
205                               logger=self.logger)
206             tpsa.start()
207             ini = datetime.datetime.utcnow().strftime("%Y-%m-%d %H
                :%M:%S.%f")
208             obj = {"ini": ini}
209             self.instantiator.psas_tt[psa['name']] = obj
210
211             tpsas.append(tpsa)
212
213         while 1:
214             avgr = 0
215             for tpsa in tpsas:
216                 remaining = tpsa.vm_status()
217                 if remaining[0] > 0:
218                     avgr += remaining[0]
219             if len(tpsas) > 0:
220                 if avgr / len(tpsas) >= 80:
221                     i.set()
222                     self.instantiator.tvdm_finishes_migration =
223                     datetime.datetime.utcnow().strftime("%Y-%m-%d
                        %H:%M:%S.%f")
224                     self.logger.info("---> PSAs migrated [event: %
                        s] \
225                     " % (str(i.isSet())))
226                     break
227
228         self.logger.info("---> waiting for handover ack to the PSC
                [event: %s \
229         ]" % (str(i.isSet())))
230         he.wait()  # wait for the handover answer to the PSC
231
232         '''
233         PSC migration
234         '''
235         self.original_psc = (self.obtain_disk_base(self.PSC['disk
            ']['location']))
236         self.generate_remote_disk(self.original_psc, self.PSC['
            disk']['type'],
```

```
                                    self.PSC['disk']['location'],
                                       self.namePSC)
        tMigration = TMigration(vm=self.namePSC,
                                sleep=2,
                                local_host=self.local_host,
                                remote_host=self.remote_host,
                                remote_port=self.remote_port,
                                logger=self.logger)
        tMigration.start()
        print ("tvdm_ini_psc puestooooooooooooo")
        self.instantiator.tvdm_ini_psc =
          datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%
             f")

        '''
        TVD migration
        '''
        try:
            nat = self.remote_info['nat']
            headers = {'Content-Type': 'application/json',
                       'Accept': 'application/json'}
            payload = {'route': nat['route']}
            url = "http://%s:%s/switch" % (str(nat['server'][0]),
                                           str(nat['server'][1]))
            self.logger.info("NAT Request URL: %s" % url)
            r = requests.post(url, json=payload, headers=headers)
            self.logger.info("NAT request Response: %s" % str(r.
               status_code))
        except Exception as e:
            self.logger.error("NAT request error\n%s" % (str(e)))

        time.sleep(1)

        self.TsendTVD.start()  # pass TVD, after init PSC
           migration

        while 1:
            if tMigration.executed:
                break
        remaining = tMigration.vm_status()
        while remaining:
            if not remaining:
                self.logger.info("No migration in progress")
            else:
                self.logger.info("remaining to migrate %s, psa: %s
                    " %
                                 (str(remaining), str(tMigration.
                                    vm)))
            time.sleep(1)
            remaining = tMigration.vm_status()
            if remaining[0] >= 80:
                # subprocess.check_call()
                break

        for tpsa in tpsas:
            tpsa.join()
```

```
287                 self.instantiator.psas_tt[tpsa.vm]["fin"] =
288                       datetime.datetime.utcnow().strftime("%Y-%m-%d %H
                          :%M:%S.%f")
289
290             self.TsendTVD.join()
291             tMigration.join()
292             self.instantiator.tvdm_fin_psc =
293                    datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M
                       :%S.%f")
294
295             for psa in self.PSAList:
296                 self.undefined_vm(psa['name'])
297
298             self.undefined_vm(self.namePSC)
299             self.logger.info("PSC migrated")
300
301
302         def undefined_vm(self, vm):
303             self.local_dom = self.conn.lookupByName(vm)
304             self.local_dom.undefine()
305
306
307         def obtain_disk_base(self, path):
308             '''
309             obtain path VM disk base
310             '''
311             try:
312                 proc = subprocess.Popen("qemu-img info " + path,
313                                         stdout=subprocess.PIPE, shell
                                            =True)
314                 (out, err) = proc.communicate()
315                 returnedValue = str(out)
316                 start = 'file: '
317                 end = '\n'
318                 disk_base = ((returnedValue.split(start))[1].split(end
                    )[0])
319
320             except subprocess.CalledProcessError as e:
321                 self.logger.info("Calledprocerr:" + e)
322
323             return disk_base
324
325         def generate_remote_disk(self, original, disk_type, path, vm):
326             """
327             Generate remote disk
328             """
329             stderr = ""
330
331             try:
332                 cmd = "[  -f " + path + "/" + vm + " ] || qemu-img
                       create -b "
333                       + original + " -f " + disk_type + " " + path
334                 stdin, stdout, stderr = self.ssh.exec_command(cmd)
335                 stdin.close()
336             except Exception as e:
337                 self.logger.info("Error to create remote disk " + str(
```

```
                e) + " "
338                             + str(stderr))
339
340     def rename_remote_disk(self, path, vm):
341
342         stderr = ""
343
344         try:
345             cmd = "mv " + path + "/" + vm + " " + path + "/" + vm
                    + "backup"
346             stdin, stdout, stderr = self.ssh.exec_command(cmd)
347             stdin.close()
348
349         except Exception as e:
350             self.logger.info("Error to create remote disk "
351                             + str(e) + " " + str(stderr))
352
353
354     def createSSHClient(self, server, port, user, password):
355
356         client = paramiko.SSHClient()
357         client.load_system_host_keys()
358         client.set_missing_host_key_policy(paramiko.AutoAddPolicy
                ())
359         client.connect(server, port, user, password)
360         return client
361
362     def execute_command(self, ssh, cmd, sudo):
363         '''
364         Create remote conection
365         '''
366         try:
367             if sudo:
368                 cmd = "sudo -S -u qemu '' %s" % cmd
369             stdin, stdout, stderr = ssh.exec_command(cmd)
370             if sudo:
371                 stdin.write('xxx\n')
372             stdin.write('x\n')
373             stdin.flush()
374
375         except socket.timeout as serr:
376             self.logger.info("Error1: %s" % serr)
377
378         except socket.error as serr:
379             print "Error2: %s" % serr
380
381         except:
382             print "ANY1"
383         return (stdout.readlines(), stderr.readlines())
384
385
386 class TMigration(threading.Thread):
387     def __init__(self, vm, sleep, local_host, remote_host,
            remote_port,
388                  logger, group=None, target=None, name=None,args
                        =(),
```

```
389                     kwargs=None, verbose=None):
390         super(TMigration, self).__init__(group=group, target=
                target,
391                                          name=name, verbose=
                                             verbose)
392         self.vm = vm
393         self.sleep = sleep
394         self.local_host = local_host
395         self.remote_host = remote_host
396         self.remote_port = remote_port
397         self.logger = logger
398         self.local_conn_string = "qemu:///system"
399         __last_bytes = -1
400         __last_time = datetime.time()
401 
402         try:
403             self.remote_conn_string = "qemu+ssh://"
404                                       + self.remote_host + ":"
405                                       + str(self.remote_port) + "/
                                          system"
406         except:
407             self.logger.info("Error Connecting remote VM " + self.
                remote_host)
408 
409         try:
410             self.conn = libvirt.open(self.local_conn_string)
411         except:
412             self.logger.info("Error Connecting to VM hypervisor "
413                              + self.local_conn_string)
414         try:
415             self.remote_conn = libvirt.open(self.
                remote_conn_string)
416         except:
417             self.logger.info("Error connecting to remote
                hypervisor: "
418                              + self.remote_conn_string)
419 
420         if (self.sleep > 0): self.logger.info(time.sleep(self.
            sleep))
421 
422         self.logger.info(
423                         "Starting migration of VM domain '" + self.
                            vm + "' from "
424                         + self.local_host + " to " + self.
                            remote_host + "...")
425         try:
426             self.local_dom = self.conn.lookupByName(self.vm)
427         except:
428             self.logger.info("VM does not exist: " + self.vm)
429         self.executed = False
430         return
431 
432     def run(self):
433 
434         try:
435             self.logger.info("RUN DE TMIGRATION")
```

```
436             self.local_dom.migrate(self.remote_conn, libvirt.
                    VIR_MIGRATE_LIVE |
437                                    libvirt.VIR_MIGRATE_COMPRESSED,
                                           None,
438                                    None, 0)
439             self.logger.info("Migration of VM domain '" + self.vm
                    + "' from "
440                              + self.local_host + " to " + self.
                                 remote_host
441                              + "triggered OK!!")
442             self.executed = True
443         except Exception as e:
444             self.logger.info("Error to migrate " + self.vm)
445             self.logger.error(str(e))
446 
447         try:
448             remaining = self.vm_status()
449             while remaining[0] != 100:
450                 if not remaining:
451                     self.logger.info("No migration in progress")
452                 else:
453                     self.logger.info("tmigration remaining to
                            migrate %, psa: "
454                                      % (str(remaining), str(self.
                                         vm)))
455                 time.sleep(1)
456                 remaining = self.vm_status()
457         except Exception as e:
458             self.logger.info("Error GETTING STATUS " + self.vm)
459             self.logger.error(str(e))
460         return
461 
462     def vm_job_stats(self):
463         if self.local_dom.info()[0] != 5:
464             return self.local_dom.jobStats()
465         else:
466             return {}
467 
468     def __update_migration_status(self):
469         dictionary = self.vm_job_stats()
470         if dict:
471             if not 'data_remaining' in dictionary:  # No migration
                        in progress
472                 if self.executed:
473                     return [100]
474                 return [-1]
475             remaining_bytes = dictionary['data_remaining']
476             total_bytes = dictionary['data_total']
477             if total_bytes == 0:
478                 if self.executed:
479                     return [100]
480                 return [-1]
481             if self.__last_bytes < 0:
482                 self.__last_bytes = remaining_bytes
483                 self.__last_time = datetime.datetime.now();
484                 eta = datetime.timedelta()
```

```
485             elif remaining_bytes == 0:
486                 self.__last_bytes = 0
487                 self.__last_time = datetime.time();
488             else:
489                 byte_rate = self.__last_bytes - remaining_bytes
490                 time_now = datetime.datetime.now()
491                 time_between_queries = time_now - self.__last_time
492                 eta_microsec = self.timedelta2microseconds(
                        time_between_queries)
493                                 * int(remaining_bytes / byte_rate)
494                 eta = self.microseconds2timedelta(eta_microsec)
495                 self.__last_time = time_now
496                 self.__last_bytes = remaining_bytes
497             percent_done = remaining_bytes / total_bytes * 100
498             return [percent_done, eta.days, eta.seconds, eta.
                    microseconds]
499         else:
500             if self.executed:
501                 return [100]
502             return [-1]
503 
504     def vm_status(self):
505         try:
506             if not self.executed:
507                 return [-1]
508             return self.__update_migration_status()
509         except Exception as e:
510             self.logger.info("vm_status vm not found %s" % (str(e.
                    message)))
511             return [100]
512 
513     def timedelta2microseconds(self, time):
514         if not time:
515             return -1
516         return (((24 * 60 * 60 * time.days) + time.seconds)) * 10
                ** 6
517                 + time.microseconds
518 
519     def time2microseconds(self, time):
520         if not time:
521             return -1
522         return (((60 * time.hours) + time.minutes) * 60 + time.
                seconds)
523                 * 1000000 + time.microseconds
524 
525     def microseconds2timedelta(self, pass_microseconds):
526         given_microseconds = int(pass_microseconds)
527         microseconds = given_microseconds % (10 ** 6)
528         rest = int(given_microseconds / 10 ** 6)
529         seconds = int(rest % (60 * 60 * 24))
530         days = int(rest / (60 * 60 * 24))
531         return datetime.timedelta(days=days, seconds=seconds,
532                                   microseconds=microseconds)
533 
534     def microseconds2time(self, given_microseconds):
535         microseconds = given_microseconds % (10 ** 6)
```

```
536         rest = int(given_microseconds / 10 ** 6)
537         seconds = int(rest % 60)
538         rest = int(rest / 60)
539         minutes = int(rest % 60)
540         rest = int(rest / 60)
541         hours = int(rest % 60)
542         return datetime.time(hours=hours, minutes=minutes, seconds
                =seconds,
543                              microseconds=microseconds)
```

B.3 Bibliography

Bibliography

[1] SECURED: https://www.secured-fp7.eu

[2] SECURED, consortium: https://www.secured-fp7.eu/about/consortium

[3] Definition of OPEN-FLOW: https://www.opennetworking.org/sdn-resources/openflow

[4] Definition QEMU: https://ca.wikipedia.org/wiki/QEMU

[5] Description of Strongswan: https://www.strongswan.org

[6] Organic Law on Data Protection: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_de_Car%C3%A1cter_Personal_de_Espa%C3%B1a

[7] OpenDayLight: https://www.opendaylight.org/

[8] Openvswitch: http://openvswitch.org/

[9] API REST: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[10] Definition of anti-phishing: https://en.wikipedia.org/wiki/Anti-phishing_software

[11] Information on the libvirt library: https://libvirt.org/migration.html

[12] Information about JSON: https://ca.wikipedia.org/wiki/JSON

[13] Definition of flags: http://whatis.techtarget.com/definition/flag

[14] Description of the Paramiko bookstore: http://www.paramiko.org

[15] Description of the libvirt library: https://libvirt.org/migration.html

[16] Infography Sustainability: https://libvirt.org/migration.html

[17] Sharelatex documentation for the relocation of the TFG: https://libvirt.org/migration.html

[18] NUC information: hhttps://en.wikipedia.org/wiki/Next_Unit_of_Computing

[19] Definition of the mobike: https://wiki.strongswan.org/projects/strongswan/wiki/MobIke