SoftFIRE

Software Defined Networks and Network Function Virtualization Testbed within FIRE+

Grant Agreement Nº 687860

D3.4
Report on First Wave of experiments on the SoftFIRE testbed

WP3 - Experimentations management

Version:	1.0.
Due Date:	January 31st, 2017
Delivery Date:	March 6th, 2017
Type:	Report (R)
Dissemination Level:	Public
Lead partner:	Ericsson
Authors:	All Partners (See List of Contributors below)
Internal reviewers:	PMC

Disclaimer

SoftFIRE has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 687860.

Version Control:

Version	Date	Author	Author's Organization	Changes
0.1	November 8′ 2016	M.Persichini, M.Calavaro	Ericsson	First draft
0.2	December 24′ 2016	L. Tomasini	TUB	Add NFV@Edge experimenter KPIs
0.3	February 5th, 2017	Susanne Kuehrer	EIT Digital	New structure of deliverable (cancelling confidential parts)
0.4	February 16th	Daniele Camignani Fabio Paglianti	Security Reply	Added results MARS experiment
0.5	February 23, 2017	Marco Miglione	Ericsson	Inserted contributions from experimenters Expose
0.6	February 27, 2017	Peter Feil	Deutsche Telekom	Added results Secgene experiment
		Serdar Vural	University of Surrey	Added results of Solid experimente
0.7	February 28, 2017	Giuseppe Carella	TUB	Added results NFV@edge
0.8	March 3, 2017	Roberto Minerva Marco Miglione Umberto Stravato	EIT Digital Ericsson	Completing chapter 2, 4 and 6, first editing of whole document
1.0	March 6, 2017	Susanne Kuehrer Fabio Paglianti Marco Miglione	EIT Digital Security Reply Ericsson	Final editing for submission

Annexes:

Nº	File Name	Title
none		

Contributors:

Contributor	Partner
Daniele Carmignani, Fabio Paglianti	SECURITY REPLY
Paolo DI FRANCESCO, Paolo CRIVELLO	LEVEL7 (Experimenter)
Marco Miglione, Umberto Stravato	ERICSSON
Harilaos Koumaras, Christos Sakkas, Anastasios Kourtis, Konstantinos Dagakis, Athanasios Drigkas, Fotis Lazarakis, Antonis Alexandridis	NCSR DEMOKRITOS (Experimenter)
Gerry Foster, Serdar Verul	UoS
Giuseppe A. Carella, Lorenzo Tomasini	TUB
Peter Feil	Deutsche Telekom
Björn Riemer	FOKUS
Roberto Minerva	EIT Digital
Susanne Kuehrer	EIT Digital

Deliverable Title: Report on First Wave of experiments on the SoftFIRE testbed

Deliverable Number: D.3.4

Keywords: Experimentation, federated testbed, first Open Call

Executive Summary:

SoftFIRE has developed a federated experimental platform aimed at the construction and experimentation of services and functionalities built on top of NFV and SDN technologies.

SoftFIRE is offering the opportunity to use the federated environment to the selected Experimenters in order to allow them to test the proposed solutions enabling a vaster ecosystem to follow up the creation of services as well as the functional extension of the platform itself.

The present report describes what has been done to set up the first Open Call of experimentation, from information spreading to Experimenter selection, Experiment execution and follow up on the obtained results.

The report contains suggestions and recommendations in terms of enhancements to the testbed or feedback for scope refinement for next wave of Open Calls.

Table of Contents

List of Figures

List of Tables

1 Introduction

This Report describes what has been done to set up, execute and follow up all experiments that were selected in the first Open Call of SoftFIRE Project and are part of Wave 1.

The document will map the KPIs to the result obtained during the Experimentation. It will report suggestions and recommendations as well to WP1 and WP2 in terms of enhancements to the testbed. In addition, feedback will be provided in order to better scope down the next waves of Open Calls.

2 Open Call preparation

The SoftFIRE project had been extensively working on creating a competitive and fair framework for the evaluation of proposals received during the first Open Call [ref to OpenCall https://www.softfire.eu/wp-content/uploads/SoftFIRE_OpenCall_20160831.pdf]. The goal has been twofold: on one hand, to attract many proposals so that the SoftFIRE project can stimulate the growth of an ecosystem of developers interested in NFV/SDN/5G; on the other hand, SoftFIRE tries to select the best proposals received from a technical and business perspective.

2.1 Open Call Principles

In order to attract a good participation to the first Open Call, the SoftFIRE project has carefully addressed the potential experimenters' domain. SoftFIRE decided that all received proposals were to be evaluated towards three general parameters:

- **Excellence** of the idea: this means to evaluate the proposed experiment novelty and its differentiation from the basic and well known possible applications. It is important to collect as many as possible original, robust and strong proposals that can demo the value of the NFV/SDN proposition with respect to evolution towards 5G.
- possible **Impact** on the field: the SoftFIRE project is looking for proposal that can have a technological or business impact on the evolution of the NFV/SDN domain in such a way to accelerate the deployment and the vast adoption of the technologies
- **Implementation**, i.e., how difficult the implementation could be and how the implementation is leveraging the functions and mechanisms offered by the platform.

SoftFIRE figured out that the NFV/SDN technology could cover two large "basins" of experimenters: the academy and the industry. In addition, there is a good number of SMEs that are interested in entering in this new potential market. For this reason, the Open Call was formulated to be appealing for all these potential experimenters.

The solicited proposals could, in fact, refer to "Scientific Excellence" (Objective 1) or "Industrial Innovation" (Objective 2), i.e., working more on the original development of new concepts or a consolidation and usage of the platform towards industrial applications.

Platform and application developments, actually, require different skills and business perspectives.

The project felt that Objective 1 and Objective 2 should have been distinguished. The distinction of two types of "objectives" was conceived in order to differentiate between platform experts prone to platform extensions and experimenters more keen to develop applications and services over a distributed platform. These two types of experiments seemed to be very different in nature especially from the business perspective. SoftFIRE decided that they had to be evaluated with slightly different criteria (more technologically oriented for the platform and more business oriented for the applications and services developments).

This approach had to be blended and mitigated with the need to be able to support the complexity of the experiments themselves. The SoftFIRE project has then sought for a tradeoff between technical complexity of the proposals and their actual implementability on the current version of the platform. This has proved a good approach, because a number of proposals were considering a “future” platform that is not possible to have currently with the state-of-art technology. There is so much need for an experimental platform in NFV/SDN/5G that researchers and practitioners would like to have all issues fixed and solved in order to start developments straight away.

SoftFIRE view on the current status of technology is less optimistic but more pragmatic. For this reasons, the project decided to include in the evaluation of the received proposals a sort of “gate” related to the actual implementation of the proposal on the platform.

Generally speaking, the project wanted to accept proposals that had a solid technical background and were feasible on the current version of the federated testbed. For these reasons a set of threshold values were imposed on certain criteria (e.g., excellence could not be lower than 7) and a system of weights in order to give a fair overall evaluation of all the merits of the proposals.

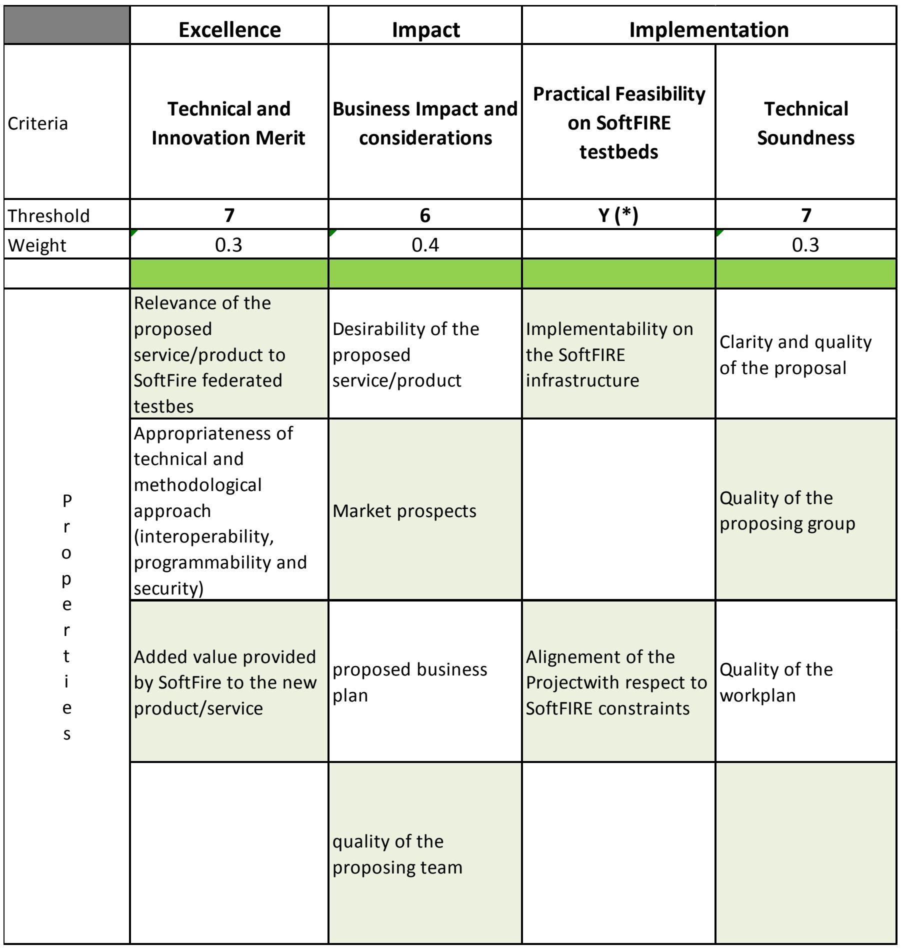
The evaluation criteria for the proposal addressing Objective 1 privileged the technicalities. The criteria are depicted in Figure 1.

	Excellence	**Impact**		**Implementation**	
Criteria	**Technical and Innovation Merit**	**Contribution to the creation of a SoftFIRE-based Ecosystem**	**Business Impact and considerations**	**Practical Feasibility on SoftFIRE testbeds**	**Technical Soundness**
Threshold	**7**	**6**	**6**	**Y (*)**	**7**
Weight	0.4	0.2	0.1		0.3
Properties	Relevance of the proposal for SoftFire federated testbes	Potential to increase knowledge at the European level and differentiate the proposition	Desirability/need of the proposed service/function	Implementability on the SoftFIRE infrastructure	Clarity and quality of the proposal
	Appropriateness of technical and methodological approach (interoperability, programmability and security)	Clarity and Quality of the Technological benefits for an European Ecosystem			Quality of the proposing group
	Originality and innovative value of the proposed features/tests and their relation to the status of the art	Contribution to standards or open interfaces		Alignement of the Projectwith respect to SoftFIRE constraints	Quality of the workplan
	Potential of exploitaiton/inclusion in SoftFire of [proposed functionalities and features (including legal/admin aspects)				Qulity of the team

Figure 1: Objective 1 evaluation criteria

This set of criteria greatly values the technical merits of the proposals over their business appeal.

Figure 2 represents the evaluation criteria for proposals addressing Objective 2.



	Excellence	Impact	Implementation	
Criteria	**Technical and Innovation Merit**	**Business Impact and considerations**	**Practical Feasibility on SoftFIRE testbeds**	**Technical Soundness**
Threshold	**7**	**6**	**Y (*)**	**7**
Weight	0.3	0.4		0.3
	🟩	🟩	🟩	🟩
Properties	Relevance of the proposed service/product to SoftFire federated testbes	Desirability of the proposed service/product	Implementability on the SoftFIRE infrastructure	Clarity and quality of the proposal
	Appropriateness of technical and methodological approach (interoperability, programmability and security)	Market prospects		Quality of the proposing group
	Added value provided by SoftFire to the new product/service	proposed business plan	Alignement of the Projectwith respect to SoftFIRE constraints	Quality of the workplan
		quality of the proposing team		

Figure 2: Objective 2 evaluation criteria

The value of the business merits had a greater weight in this second criteria set.

2.2 The Open Call Objectives

The first Open Call of the SoftFIRE project focused on interworking and interoperability in order to assess the effort of the federated testbeds in creating a single environment for experimentation. The proposed work plan of the experiments should address platform enhancements, services offered by the platform or applications running on the platform that show and leverage the distribution of functionalities over different testbeds.

2.3 The Designed Evaluation Process

The Evaluation process has been designed in order to be fair and clear for all the participant of the Open Call. In addition, due to time constraints of the project (three Open Calls in 24 months) there was a clear requirement to have a smooth and fast selection process in order to keep up with the stringent timing of the project.

The adopted evaluation process is represented in Figure 3

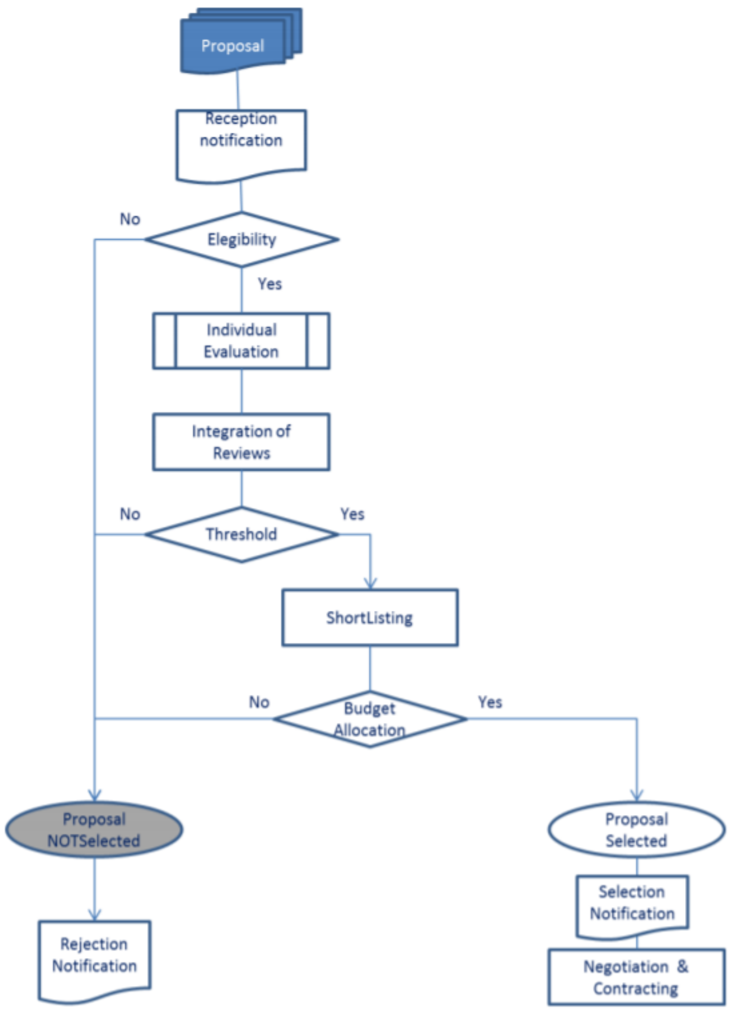


Figure 3: the Evaluation Process

The process was very straightforward and neat:

- Receipt notification (projects applying after the closing date were rejected);
- Eligibility, experiments proposed by not EU recognized entities were to be rejected;
- Individual Review by domain Experts;
- Grouping and Shortlisting;
- Budget Allocation;
- Notification of Acceptance or Rejection.

Further information about how the evaluation has been carried out are reported in section 4.

2.4 Open Call content definition

2.4.1. The Intended Open Call objectives for the SoftFIRE project perspective

The expected experiments should have addressed at least one of the following objectives:

- OBJECTIVE 1, the development and validation of extensions to the current federated testbed or of new functionalities thereof. For example:
    - testbed enhancements in terms of orchestration, control or virtualization capabilities and their real-world evaluation over the federated infrastructure;
    - architectural extensions based on NFV/SDN or new functionalities, complying with the 5G notion;
    - Experiments enabling benchmarking of the NFV/SDN components used in SoftFIRE;
    - Extensions to SoftFIRE infrastructure by using open source SDN/NFV components, interoperable with the existing FIRE control and management tools adopted by SoftFIRE.
- OBJECTIVE 2, the production and/or validation of new services, systems and applications that benefit of network programmability and NFV and SDN technologies more generally.

The intention was to stimulate and receive innovative ideas on how to improve the platform and how to use it to solve important research and industrial issues.

2.4.2. Training of Experimenters

It was clear from the beginning of the project that Experimenters would not have a detailed knowledge of the platform (that was under consolidation during the Open Call definition). So it was important to elaborate a process for easing the process of training and usage of the platform. This process was based on two major pillars for supporting the training of Experimenters:

- The preparation of documentation;
- On-site support and training.

The project has put a lot of effort in creating appropriate documentation focusing on two important aspect of the platform: a) its architecture and organization, and b) its programming.

The documentation related to the platform has been revisited and update frequently due to the extension and evolution of the platform. However, it has always been an important point for the SoftFIRE project. The First Open Call has been issued with an accompany document called "Handbook: Guidelines and Rules for on-demand access to the SoftFIRE Testbed document" (it is available here:

https://www.softfire.eu/wp-content/uploads/SoftFIRE-D3-2-Handbook-Guidelines-and-rules.pdf) [(SoftFIRE, 2016, April)].

It contained information about the used infrastructure, the software architecture and supported functions, the mechanisms to be used to program the platform. In addition, some guidelines on how to access and use the platform were also provided so that the programming process could have been executed efficiently.

In order to provide a hands-on possibility, the SoftFIRE project scheduled an InterOpTest, in which the Project Partners showed the results achieved in the federation of the testbeds and as well as demos and examples on how to use the platform. This also offered the opportunity to interested third parties (some of them were present at the InterOpTest to familiarize with the platform in order to understand its current functions and how to use them to develop applications and services. Further details about the InterOpTest can be found in chapter 3.1.

The need for a good training is a requirement that the project has learnt very quickly. Documentation alone will not work if there is not an effort in providing Face-to-Face training.

2.5 First Open Call Information

This was the official Open Call information for the potential experimenters published on the SoftFIRE website (https://www.softfire.eu/open-calls/first-open-call/):

Call title: First SoftFIRE Open Call for Experiments

Total Budget: € 500,000

Maximum Funding per proposal: € 50,000

Type of participants: The typical profile of participants is academics, industrial or SMEs active in the domain of NFV/SDN and 5G research and applications that need to run experiments to test, evaluate and optimize their solutions and applications. The rules of participation are indicated in SoftFIRE First Open Call: http://www.softfire.eu/open-calls/

Duration of the experiment: The maximum allowed duration of each experiment is 3 Months

Language of the proposal: English

Proposal submission: online submission through the portal

Call deadline: 20 September 2016 at 17:00h CET (Brussels time)

Notification of Acceptance: 14 October 2016

The experimenters were requested to send their proposals using the template provided by the project. The chosen means of interaction with experimenters is the Web portal in order to create a sort of habit (the community could refer to it for finding news about the SoftFIRE news and initiatives) and to acquaint the experimenters to a few authorization mechanisms implemented in the web portal and to be used to access the Federated Testbed. From the call, it is clear that the SoftFIRE project aimed at many small projects to be executed in a short period of time. In fact, the experiment time frame for the execution of the project is fixed to maximum three months. With this proposition, the project aimed at providing to several organizations a chance to work on a distributed platform for implementing new solutions in the realm of NFV/SDN/5G. This is also a means to involve in these developments the largest possible community.

2.6 Open Call information spread

The success of the Open Call clearly depended on the quality of the proposition. However advertising played a fundamental role in attracting interested parties.

The main vehicle for promotion was surely the project’s web portal (that had to have a quick start as well in order to reach a good level of visibility). The other main channel of promotion was the use of EU Community mailing list and advertisements.

The project's members also distribute the information about the open call to their company or personal networks so that a large community could be reached. In addition, Social Media like LinkedIn and Twitter was used to spread the voice.

2.6.1. Web Portal

The Web portal has been extensively used for advertising the Open Call. Actually the visits to the portal has peaks reaching around 300 hits per day at the time of i) the Inter-Op Test public demonstration session and ii) the Open Call kick-off. For more information about the Web Portal statistics the interested reader could refer to [(SoftFIRE, D4.1: WP4 Dissemination Report, January 2017)].

The project intends to use the web portal as its main means to inform and interacts with potential experimenters also in the future.

2.6.2. Other Channels

The Project has leveraged the possibilities offered by the several events and initiative endeavoured by the EU Commission to promote its projects to the larger audience possible. SoftFIRE often participated to events and meetings in this context creating links and interest for the project. In addition, the FIRE Initiative advertises Open Calls of its projects to the wider possible audience, using both the newsletter and also the Twitter. SoftFIRE has benefitted from this support, too, and it has reached a good response in terms of interest and participation.

In addition, many members of the project are also active members of the NFV/SDN/5G community, and the project has been promoted to standard organizations, communities and practitioners on an individual base.

Social Media was also used in an extensive way, setting up a LinkedIn account for the project; in addition, many project partners used their Twitter accounts, e.g., EIT Digital, to spread the voice about the SoftFIRE Open Call.

3 Creating the community

A consistent part of the budget of the SoftFIRE project is devoted to Third Parties. The purpose is to create an ecosystem of programmers around the NFV/SDN/5G technologies offered by the project. There is a strong need to aggregate this community because in Europe there are many entities interested and potentially capable of providing solutions in this realm. There are other testbeds operated in Europe, but SoftFIRE is unique in terms of openness and federation. In fact, available testbeds are running under the responsibility of Telecom Operators or Vendors and are used for specific business purposes. The goal of SoftFIRE is to help a large community of companies and people interested in developments in accessing to NFV/SDN/5G capabilities without necessarily creating a strong business link with a specific actor of the sector. In fact, universities and SMEs were quite responsive to the first SoftFIRE Open Call.

This could be considered a success in itself, because SoftFIRE allows experimenters to freely use the Federated Testbed for their own experiments (either aiming at scientific excellence or business goals) without forcing them towards a specific direction strongly influenced by business requirements. The experimenters can identify a technological problem that needs to be solved or they can think about developing a service/application with their own business model in mind.

The First Open Call and the InterOpTest have given some indication along this path. The major directions that SoftFIRE is following in order to create the community are:

- to provide many Open Calls creating a set of waves of experimentation. This means that the usage of the platform will be offered to many interested entities more than focusing on a few large experimentations;
- to offer access to the platform functionalities during events in such a way to familiarize potential entities to embark on experimentations
- to offer access to the platform on demand and for a fee to those entities that want to develop their own solutions within a different timeframe.

The SoftFIRE project has scheduled this set of activities:

- 3 Open Calls for Experimenters
- Several events will be organized:
  - InterOp Test
  - 2 Hackathons
  - Final Challenge/Hackathon

This section reports about the activities and results achieved so far in creating and advertising events as well as the Federated Testbed and the activities supporting the potential experimenters, while section 5 describes the work done by the single experiments carried out during the first Open Call execution (the first wave of experimentation).

3.1 InterOpTest

The InterOpTest was held in Berlin on September 30th 2016, it was launched in parallel with the release of the first Open Call. The intended goal of the event was to publicly prove that the Federated Platform was up and working (interoperability of the different individual testbeds) and it was possible to program it.

In order to demonstrate the achievement, a general description of the Federated Testbed (objectives and high level architecture) has been presented, then a tutorial about how to program the infrastructure was provided and eventually a set of services were introduced as well as the security framework.

The interested reader can find the agenda of the event and the related multimedia material being produced available at following link for Open InterOpTest: https://www.softfire.eu/events/open-interoptest/

The event offered the possibility to be present in person or to participate remotely. A global audience of more than 40 people was registered for this event with many people attending in person. This was a good recognition of the interest received by this initiative.

The Introduction to that day has provided the main goals of the SoftFIRE project and the current architecture under development:
https://www.softfire.eu/wp-content/uploads/SoftFIRE_Intro-20160930.pdf .

This presentation has been followed by a general description of the first Open Call aiming at providing some indications about the participation and execution of the first Open Call itself https://www.softfire.eu/wp-content/uploads/SoftFIRE_InterOpTest_Open-Calls_Berlin_300916.pdf

The next presentation was about the programming of the platform i.e., a hint on the supported software lifecycle and the way programmers should access the Federated Testbed functionalities: https://www.softfire.eu/wp-content/uploads/SoftFIRE_InterOpTest_tutorial_on_programming_the_platform_300916.pdf ,

Then a set of Demos has been provided in order to

(a) introduce the audience to the intricacies of the network supporting the Federated testbed https://www.softfire.eu/wp-content/uploads/SoftFIRE_Network-Structure.pdf and

(b) show how to create a complex IMS communication service https://www.softfire.eu/wp-content/uploads/SoftFIRE-IMS-demonstration.pdf and

(c) the security framework https://www.softfire.eu/wp-content/uploads/SoftFIRE_InterOpTest_SecurityDemo.pdf.

The event then provided enough time for Q&A sessions (also with remote users) in order to sort out doubts and provide indication on Open Call and next steps of the project.

The attendees also created a direct link with many of the SoftFIRE partners and these were instrumental to support further discussions and create more durable relationships with potential users of the platform.

The educational value of the event was clear to all SoftFIRE partners and the need to "educate" potential experimenters to the features of the Federated Testbed already emerged during the several discussions.

Education to the distributed programming over a virtualized platform is an important issue that the project has encountered along its activities. There is a need to provide hands-on possibilities to potential experiments so that they can speed up the learning curve for making effective use of the platform. One consideration is then to use scheduled events in order to make them also relevant to the educational path. For instance, future Hackathons could be more valuable if they are executed before the end of Open Call because the potential experiments could understand better the working of the platform and could use this knowledge to tune-up their proposals in order to better fit with the platform capabilities. This approach has to be fully exploited by the project in order to create more effective events.

3.2 Forum

3.2.1. Forum as a Service

Redmine portal (http://www.redmine.org/) has been structured in several sections to support the different moments of collaboration with external team during Open Call and Hackathons.

In details, the consortium has decided to create different sub-forums where the experimenters can directly access and exchange information among them or with the infrastructure providers.

Ideally, it shall facilitate the knowledge sharing and transfer to following waves experimenters. Also, this forum section has been structured to support the creation a community around SoftFIRE platform.

In all sub-forums a push notification system has been implemented, so in the same moment a user creates any post, the person in charge to moderate any thread is automatically informed of the post and based on the urgency judged if the post should be answered immediately or not, also redirecting it to another forum user.

3.2.2. List of Forum Topics

This is the structure of the entire forum system.

We have two sub-forums, here listed:

- General Forum
  Generic threads about the SoftFIRE platform. Open Thread Discussion for all (Consortium Partners and Experimenters).
- Community Rules And Policies
  Forum Rules and Guidelines for the all the users. Open Thread Discussion for all (Consortium Partners and Experimenters).

In addition, to properly support the use of the SoftFIRE platform by the experimenters during the Open Call, we have set up the following three sub-forums, monitored directly by defined Consortium Partner(s):

- jFed
  Technical and dedicated support for jFed. Monitored by FOKUS / TUB Team.
- Design – OpenBaton
  Technical and dedicated support for OpenBaton. Monitored by FOKUS / TUB Team

- Testbed
  We have set a sub-forum for each Testbed to support experimenters during the Open Call. Each sub-forum is monitored by the relevant Testbed owner.

Last, but not least, we have set also a supporting forum for Open Call Winners:

- OpenCall
  In this one, several sections has been created to support the Waves – one for each Open Call Wave. In the proper thread, the Project Coordinators publish the info of each winner per Wave, so the proper Consortium Partner can provide the access to them in an automatic way.

3.3 Support for experimenters

The SoftFIRE project has devoted particular attention to the relationships with experimenters. The Federated Testbed, even if robustness and resilience is sought for, is still in an early experimental stage. For this reason, the experimenters have been considered as partners sharing an objective. The support for experimentation was originally based on two pillars:

- A rich and extensive documentation (the handbooks). This required a great effort to the project, producing adequate documentation while the Federated Testbed was still under development was a daunting challenge;
- A ticketing process, i.e., the issues and bugs identified by the experimenters should be reported on the Redmine tool and the project representatives should sort the issue out.

Soon, in the experimentation phase, it emerged that these pillars were needed but not sufficient to support the experimenters. Two actions have been put in place in order to better allow the experimenters to carry out their objectives under "the spirit of cooperation" mentioned before:

- The project started a Mentorship program, i.e., each experiment has a unique point of contact. This point of contact is a SoftFIRE representative that interacts with the related experimenters involved in the experiment in order to get feedback, spot issues in advance or to guide them to use the Federated Platform;
- The Interactive Sessions, i.e., the interaction between experts of the Federated Platform and the experiment's group in such a way to get a description of the intended software architecture of the experiment and to offer guidelines and hints on how properly create packages to run on the Federated testbed.

These two actions proved very effective, the Interactive sessions mechanisms has been successful to allow some experiments' group stuck in the intricacies of the Federated Testbed to quickly move on to an effective development phase. On the other side, the Mentorship program was able to identify the platform issues very much in advance, and to create a cooperative atmosphere with the experimenters. Actually, some issues identified by experiments that were more advanced in the development stage have been solved and passed to other experiments in order to quickly move on the development. During the contractual phase of the first Open Call, the SoftFIRE project has requested the experimenters to define, together with the Mentor, a set of meaningful KPIs to be evaluated in order to understand if the experiment was reaching its goals. This KPI definition process has been interactive and the Mentor has contributed to it in order to mediate the results with the actual capabilities of the platform. At the beginning, the Mentor was “interpreted” by some experimenters as a controlling figure. This is not the case, the Mentor was established in order to be a facilitator and to help the experiment to better use the platform and achieve its goals.

On the other side, this approach has requested additional effort on the Project members, with particular impact on the people that have a deep knowledge of OpenBaton and other components of the platform. This evidently had also an impact on the work about consolidation carried out by the same people. For future Open Calls, the SoftFIRE project will try to have a wider set of skills available within the partners in order to share more evenly the burden of mentoring.

The relationships created with experimenters have created links that will last between different groups and it will be exploited by SoftFIRE to nurture the ecosystem.

The project has also tried to create links and interactions with groups that submitted proposals that were not accepted. A first action was to provide feedback to the proponents in order to show them that we have carefully considered the proposal with due interest, fairness and respect. The other point was also to suggest a few strong points s that the proposal could also be improved.

This is an attempt to tempt people to resubmit proposals for the next Open Calls. These proposals could be reshaped and improved or simply re-proposed for consideration because the SoftFIRE Testbed has introduced new functions that could allow now the implementation of the experiment.

4 Review and selection of Experiments

4.1 Proposal collection First Open Call

The first Open Call has gained attention from the NFV/SDN/5G communities and gathered several good proposals, more in detail SoftFIRE got 26 proposals. Among these, 12 proposals were addressing Scientific Excellence, 13 proposals Industrial Innovation and 1 was referring to both.

The Applicants were 50% Companies and 50% Academia.

This is an important confirmation for the approach adopted by the project about promotion of academic and industrial innovation.

Proposals came from nine different countries, showing an interest well distributed around Europe. The Countries were:

- Cyprus (1),
- Finland (1),
- Greece (11),
- Italy (5),
- Netherlands (1),
- Serbia (2),
- Sweden (1),
- Spain (3),
- UK (1)

For next Open Calls, the project will intensify its promotion in addressing in particular important realities of the NFV/SDN communities in Germany, France and other important countries in this area to attract even more interesting proposals that could support the value of the SoftFIRE federated testbeds.

4.2 The Evaluation Process

The Evaluation process has strictly followed the process highlighted in section 2.3. Particular attention has been paid to the following issues:

1) To deeply analyse the experiment proposals in order to ensure that the chosen ones were executable on the current version of the federated testbed
2) To focus on requested functionalities (network and SDN) in order to ensure that they were available and compatible with the intended usage of the experimenters.
3) To guarantee a high quality and excellence of the proposals.
4) To promote and evaluate some business perspectives of the experiments.

Actually, points from 1 to 3 have been achieved. With respect to point 4 it may be said that there is still some work and experimentation to be done before having a clearer vision of the business impact of the single services and applications. The community up to now is still technically oriented.

Experts of the problem domain (internal and external to the project) have been selected in order to form a Jury and to evaluate the received proposals. Even the proposals that were judged not feasible on the current version of the Federated Testbed have received feedback. As said careful attention has been paid to the implementability of the platform and for this reason the internal reviewers have strongly supported the external reviewers. The details of the evaluation process are confidential and will be reported in an Annex to Del. 5.3 Project Management and Activity Report.

Some projects assumed the availability and the uniform distribution of functionalities on the whole platform. This was a misinterpretation that the SoftFIRE Project will take into account and avoid for the next Open Calls. The availability of a fully-fledged NFV/SDN/5G platform is somehow unrealistic considering the current state of the art solutions developed by the project and also in the larger NFV/SDN/5G community. However, this points to an interesting necessity of the entire community: to have a large platform that could be used by many organizations in order to develop and improve NFV/SDN/5G solutions. Only major organizations can have the strength to create and run such a large platform. The existence of an open and growing platform such as SoftFIRE has gained attention and interest from many organization that have the need to scale up their developments.

The selected projects were:

1) EXPOSE,
2) MARS,
3) NFV@EDGE,
4) SECGENE
5) SOLID
6) Tracking FLE

These experiments cover a good spectrum of possibilities ranging from network related improvements up to the possibility to automatize the software production on this kind of platforms. This was quite satisfactory from the SoftFIRE project perspective. These projects are presented and discussed in section 5.

The SoftFIRE project worked out with the selected experiments a shared and agreed document for each individual project stating the overall experiment goals and a set of verifiable KPIs needed to evaluate the results of the experiments at the end of the execution period. All documents related to the contract with the experimenters are confidential and will also be reported in the Annex to Del. 5.3 Project Management and Activity Report.

5 Experiment execution

The project has instantiated a mentorship program for supporting the experiments by assigning one mentor to each project. Further the mentor guarantees a proper handling of issues and provided constant support to experimenters as well as some educational support.

Sections 2 and 4 have shown how the experimentations have been prepared, section 3 has described how the community has been created. In section 5 now, a description of each of the selected experiments is provided. The following paragraphs are giving the information on the different experiments and depending on the nature of the experiment some parts might be longer than others. The reason is that experiments are very different in nature and they cover different topics.

Each experimenter is reporting a description of the experiment and its setup into SoftFIRE platform and also relevant results obtained in terms of performance analysis versus planned KPIs. Also feedbacks and lessons learnt on the single experiment are reported.

In Table 1 below the six selected experiment in the order how they will be presented in the subchapter of Section 5 indicating also the assigned mentors:

Proposal short name	**Proposal title**	**Affiliate**	**Mentor**
Expose	Extension of the Federated Platform with NFV/SDN-compatible Emulator of Satellite Communication systems	National Centre of Scientific Research "Demokritos"	ERICSSON
MARS	Managing Attacks from Remote Sources	Level7 S.r.l.	Security Reply
NFV@EDGE	Network Functions Virtualization at the Edge of the Network	Politecnico di Torino, Dept. of Computer and Control Engineering	TUB
SECGENE	SEmantics driven Code GENEration for 5G networking experimentation	University of Nis, Faculty of Electronics Engineering	DT Lab
SOLID	Softfire OffLoadIng	Gridnet S.A.	TIM/U Surrey
Tracking FLE	A real-time edge tracking service for video analytics-based applications	Fujitsu Labs of Europe Ltd.	U Surrey

Table 1: The six selected experiments

During the execution of the experiments the SoftFIRE project was expecting (especially towards the end of the experimentation period) a flooding effect. Many implementations executed in parallel may have caused issues to the Federated Testbed. This “effect” did not materialize. The main reason was that Mentors were trying to optimize the resource usage and to encourage the experimenters to take advantage of the availability of the platform in certain periods. In spite of this careful optimization of usage, some problems emerged in the management of the resources:

- Some experimenters tend to use the virtual machines on individual testbed they are used to interact with. This means that some Testbed were more used than others. A better support for registration and optimization of used resources is needed
- The number of "orphan" virtual machines (i.e., virtual machines instantiated by a project and not stopped and deleted) was incrementally growing. This event associated with the previous one was critical for some testbed. A garbage collection function would be extremely useful in order to help to solve the problem.

One suggestion for the next experiments execution is to implement a naming convention for processes owned by a single experiment in order to be able to terminate the virtual machine that remain active even when an experiment has finished or has been concluded.

Another issue that was important to tackle was the "bug phenomena": at the very beginning of the experimentation phase, even after testing of the Federated Testbed, having different experiments accessing the platform from several areas have been instrumental to reveal some bugs. The Mentor and the platform experts have been working very hard in order to reduce the unavailability time of the platform, but, inevitably, these issues have impacted some experiments. However, the relationship between experimenters and Mentor has been useful in order to explain to experimenters the nature of problems and to create a cooperative attitude between experimenters and SoftFIRE members. This case was particularly important to alleviate the issues caused by the incompatibility of jFED/FITeagle and the distributed functions offered by the Federated Testbed. After working on the situation, it was decided to offer to experimenters the possibility to directly access the OpenBaton middleware in order to instantiate the different distributed virtual machines. This solution allowed the experiments to better understand the structure of the platform and to work on the distribution of the platform.

In some cases, some power failure caused damages and issues to a particular testbed, but the strong relationship (and the hard work of the SoftFIRE team) allowed to fix the situation and to complete in due time the experiment.

These insights have been very valuable to the SoftFIRE team because hands-on experience has offered very important feedbacks. The team is now consolidating the current version of the platform, it is testing it extensively and it is increasing the number of available functionalities in order to provide to future experimenters more capacity, more programmable functionalities and a richer platform to work on. This experience is also important because it pushed the project into deeper consolidation of the entire Federated Testbed and this is a value for the consortium and the ecosystem around it.

5.1 Experiment "Expose" – Demokritos (Mentor: Ericsson)

5.1.1. Introduction

SDN and NFV are considered a fundamental component in the 5G landscape, it is widely recognized that 5G networks will be software-driven and most components of future heterogeneous 5G architectures should be capable to support software-network technologies. At the same time, ubiquitous broadband connectivity, extended to rural and low-density areas, is recognized as key requirement for 5G. To that end, the role of satellite networks in tomorrow's communication landscape is indeed irreplaceable. The new generation of satellites, using diverse technologies and configurations (e.g., use of Ka, Q/V-band High Throughput Satellites (HTS), LEO/MEO (Low Earth Orbit/Medium Earth Orbit) constellations) are offering high capacity and ubiquitous connectivity under all circumstances and all locations, with uniform QoS and inherent wide-area broadcast capabilities. That is why satellites are considered an essential element of future 5G infrastructures

The aim of this deliverable is the development and validation of the necessary extensions to the current SoftFIRE federated testbed, in order to enhance it with the capability to execute experiments involving satellite communication systems. EXPOSE has also provided a satellite system emulation platform, whose components will be virtualised and suitable to run on the SoftFIRE testbed. The proposed architectural extension is aligned with the 5G notion, which considers satellite and terrestrial flexible integration especially for backhauling purposes. The extension and upgrade has been performed in an interoperable way with the existing FIRE control and management tools adopted by SoftFIRE in order to allow the execution of experiments over the upgraded/extended SoftFIRE federated infrastructure.

Upon the extension of the SoftFIRE platform, this deliverable describes two experiments for the evaluation and validation of relative use-cases associated with federated satellite/terrestrial SDN/NFV-capable networks. EXPOSE will also provide a satellite system emulation platform, whose components will be virtualised and suitable to run on the SoftFIRE testbed.

5.1.2. Overall description of the experiment purpose

The European projects ESA CloudSat and H2020 VITAL have been focused on the concept of integrating satellite components in 5G software-based networks by integrating architectures involving SDN/NFV-enabled satellite/terrestrial networks. The combination of satellite and terrestrial components to form a single/integrated telecom network has been regarded for long as an integral part of 5G future networks.

In the framework of these projects, an SDN/NFV-capable satellite communication emulator, based on standard DVB-RCS and DVB-S2 technologies has been designed, developed and validated. The aim of this project is to adapt and integrate this open source emulator into the current SoftFIRE federated testbed, so that the latter can accommodate experiments involving satellite communications.

The proposed architectural extension is aligned with the 5G notions, which considers satellite and terrestrial flexible integration especially for backhauling purposes. The extension and upgrade will be performed in an interoperable way with the existing FIRE control and management tools adopted by SoftFIRE in order to allow the execution of experiments for the real world evaluation over the upgraded/extended SoftFIRE federated infrastructure.

Overall, the proposal has the following Technical/Scientific Development Objectives:

Objective 1: Extension and adaptation of the VITAL/CloudSat NFV/SDN compatible satellite communication systems emulator to fulfil SoftFIRE technical and operational requirements

Objective 2: Integration of the satellite communication systems emulator with the SoftFIRE federated testbed and especially with Open Baton – NFVO of the SoftFIRE testbed in terms of orchestration, control and virtualization capabilities

Objective 3: Execution of experiments towards the evaluation and validation of industrial benchmarks and real use cases of 5G, both operational and analytical over the enhanced/extended SoftFIRE infrastructure.

Objective 4: Interpretation of the results in comparison to the expected 5G performances, providing feedback to the federated platform reliability, calibrating the platform functionality and providing guidelines for future experimentation on the federated platform

5.1.3. Experimental set-up over SOFTFIRE platform

The planned experiments to be executed on the federated platform are designed based on the additional technical capacity offered to the SoftFIRE platform, allowing experimenters to run experiments related to the inclusion of satellite components into future federated virtualized networks in the 5G context.

The proposed NFV/SDN-compatible satellite communication emulator, shown on Figure 4 provides an easy and flexible way to emulate satellite communication systems and it comprises of three different software components, namely: the Satellite Terminal (ST), the Satellite Emulator (SE) and the Gateway (GW).

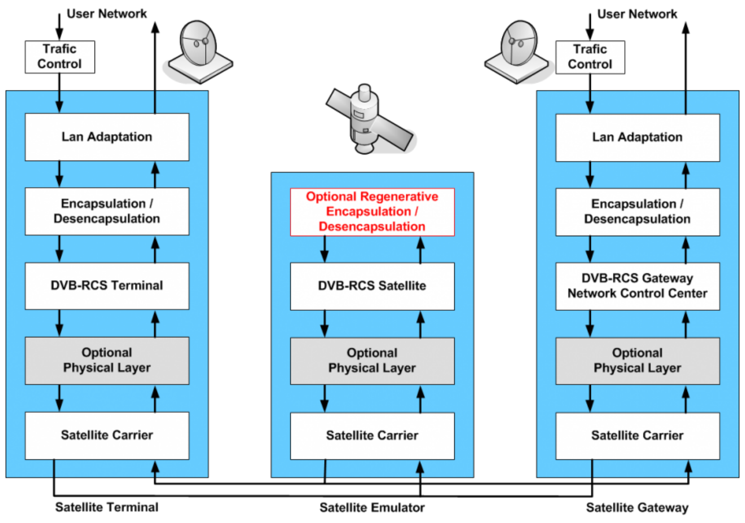


Figure 4: Description of the core of the access gateway mapping in the proposed NFV/SDN-compatible emulator

Each of these software components can run on a different PC or all in one PC. The ST emulates all the up and down link functions of a DVB-S2/RCS terminal and supports various en/decapsulation schemes. A terminal (e.g. a laptop) can be connected to it and access the services installed in the gateway. The SE is able to emulate either a transparent or regenerative satellite in combination with different encapsulation schemes depending on the payload type, up/return link standard and installed plugins. Furthermore, the SE emulates a real satellite link, including delay, signal distortions, link budget etc. The GW emulates all the functions of a DVB-S2/RCS gateway. Apart from the DVB-S2 uplink, it implements the encapsulator and the DVB-RCS functions with the required multiplexing features. An external server can be connected to the GW, which hosts the provided services, e.g. a video server. Each of the above software components will be provided as VNFs ready for instantiation in the SoftFIRE testbed.

The main scope of the emulator is to provide a research and engineering tool to validate access and network innovative functionalities in satellite networks, related to the area of 5G, utilizing NFV/SDN technologies. The emulation engine mimics the satellite DVB-S2/RCS communication system and it is based on the Open-source OpenSAND software that was initially developed by Thales Alenia Space, currently promoted by CNES and maintained by Vivéris. In the frame of CloudSat and VITAL projects, OpenSAND has been enhanced to support SDN/NFV services, i.e. allow VNF instantiation within the satellite network and also enable SDN-based network control and this open source enhanced version is proposed for extension to the SoftFIRE platform. The proposed emulator can provide measurement points and analysis tools for performance evaluation. Moreover, it can ensure interconnection with real SDN/NFV-enabled terrestrial networks and applications for demonstration purposes.

For the sake of clarity, a single Satellite Network Operator is considered, which provides a GEO satcom service via a transparent satellite. We will follow the SDN approach to achieve programmability in the provided network infrastructure and adopt the NFV concept to investigate the capability to insert virtual network services on-demand.

Having in mind the requirements expressed in the previous section, the EXPOSE experimental testbed includes:

- Satcom emulator platform, based on OpenSAND (based on DVB-RCS and DVB-S2)
- SDN programmable network segment (Openflow-enabled) of SOFTFIRE platform.
- Openstack infrastructure for VNF hosting provided by SOFTFIRE platform.

For the management and the orchestration of the infrastructure resources, the following two layers are provided:

- A management layer for managing the IT (Openstack) infrastructure as well as the SDN network
- An orchestration layer, provided by OPENBATON Orchestrator, for orchestrating the IT and network resources, and for interconnecting virtual functions to achieve service chaining.

The physical network architecture is provided in following Figure 5:

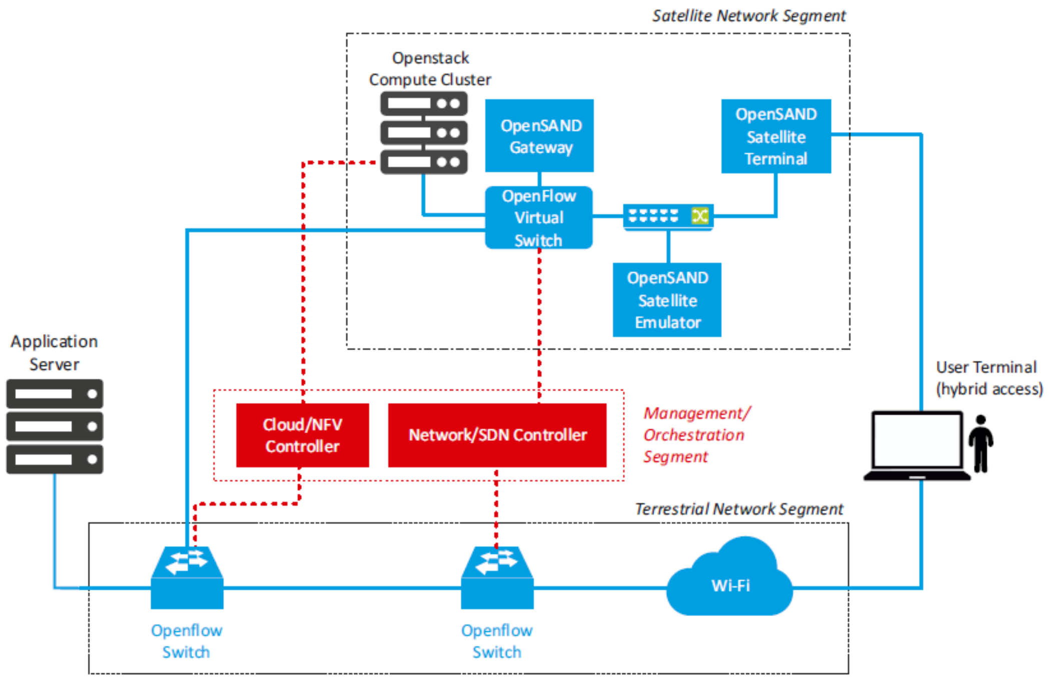

Satellite Network Segment
Openstack Compute Cluster
OpenSAND Gateway
OpenSAND Satellite Terminal
OpenFlow Virtual Switch
OpenSAND Satellite Emulator
Application Server
User Terminal (hybrid access)
Cloud/NFV Controller
Network/SDN Controller
Management/ Orchestration Segment
Terrestrial Network Segment
Openflow Switch
Openflow Switch
Wi-Fi

Figure 5: Physical Network architecture of the EXPOSE experimental setup over SOFTFIRE platform.

Figure 5 above presents an architectural view of the physical testbed architecture for the EXPOSE experiments that satisfies the aforementioned requirements. The main elements composing the architecture are the cloud infrastructure, where the virtual network appliances are running, the SDN-compatible terrestrial network and the satellite network emulator.
The virtual resources provided in EXPOSE project include virtual machines (VMs), which comprise virtualized network functions (VNFs). These are combined with other virtualized network resources and/or physical resources in order to create the virtual networks. Resource virtualization aims at better utilization of the underlying infrastructure in terms of (i) reusing a single physical or logical resource for multiple other network instances, and (ii) aggregating multiple resources in order to optimize resource usage.

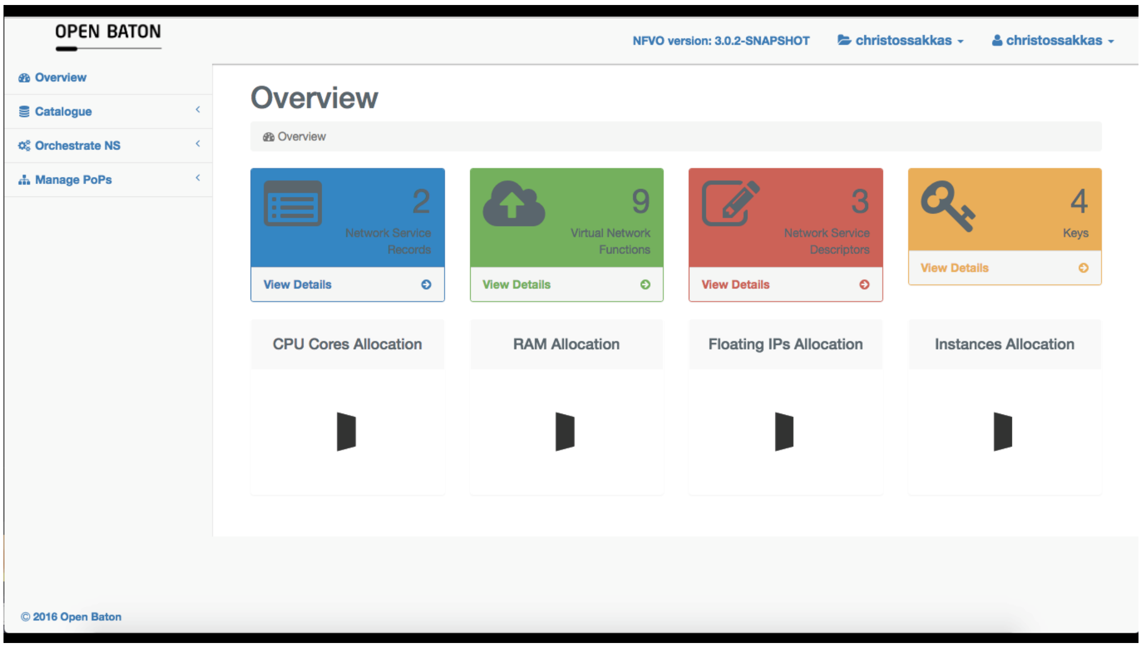


Figure 6: Open Baton orchestrator of the deployed experiment

In order to manage both virtual and physical resources effectively, an effective orchestration and network management system is used based on OpenBaton orchestrator, which is provided by the SOFTFIRE platform, as shown in Figure 6.

5.1.4. Experimentation and performance analysis

The experiments to be executed in EXPOSE are as follows:

➢ **Experiment #1: Hybrid media distribution network as-a-Service**

This scenario focuses on the federation of satellite and terrestrial domains and the provision of a hybrid satellite/terrestrial access network slice to a media service provider for content distribution. Hybrid distribution of digital media, combining satellite broadcast and terrestrial IP, is a scenario which is gaining increasing attention during the last years, due to the fact that it brings together the best of both worlds: high-bitrate and high-quality 2D/3D broadcast content, coupled with interactive personalized services.

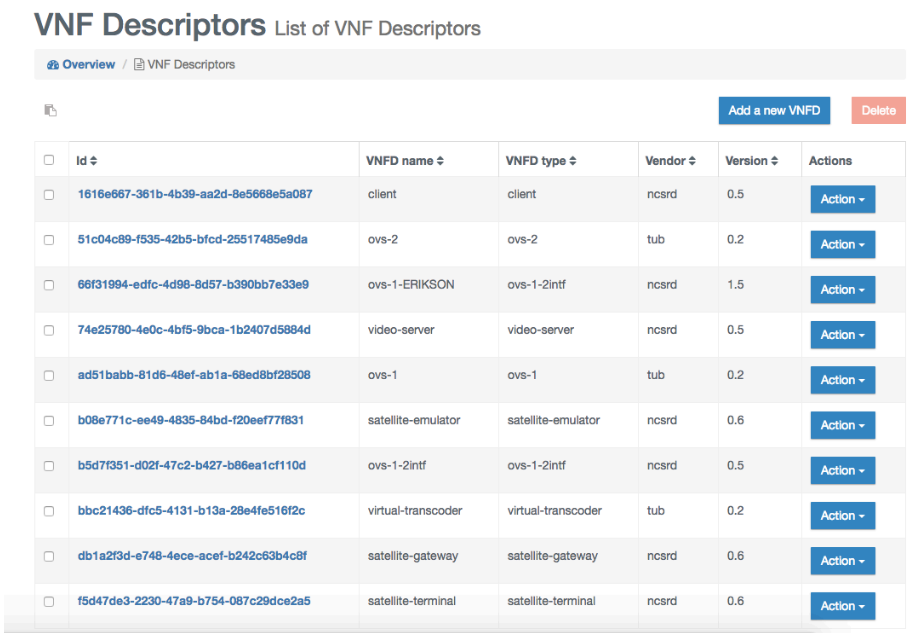
➢ **Experiment #2: Federated terrestrial-satellite VPN**

This experiment is mostly oriented to enterprise or institutional use and assumes a customer with several distributed Points of Presence (PoPs) (such as the SoftFIRE testbed), e.g. headquarters, branches, remote offices, mobile units etc which need to be interconnected into an integrated corporate Virtual Private Network (VPN). It is also assumed that some of the PoPs are outside terrestrial network coverage, for example in isolated areas or in long-haul routes (ships, airplanes etc.). For this reason, the VPN needs to encompass both the terrestrial and the satellite infrastructures in order to cover all PoPs. Satellite connectivity is not required only when terrestrial coverage is missing, but also in cases when a backup link is required for redundancy, when the availability requirements are strict (e.g. in mission critical applications etc.)

➢ **Experiment #3: Dynamic Satellite backhauling with edge processing**

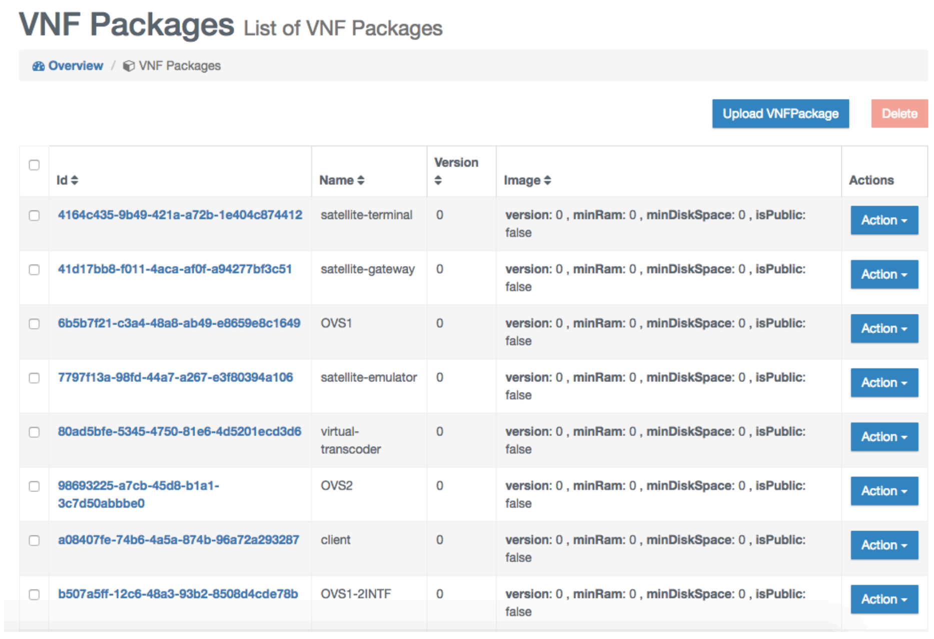
The dynamic backhauling with edge processing as-a-Service scenario investigates the dynamic extension of terrestrial networks via satellite links, in cases where terrestrial coverage is inadequate. Beyond allocating capacity on-demand and providing the necessary QoS per service, it becomes possible to also deploy instances of specific services of the terrestrial network, such as LTE eNodeB components as VNFs on the satellite access segment. This is the concept of satellite edge processing, which is in line with the emerging paradigm of Mobile Ede Computing (MEC).

For the execution of the two aforementioned experiments a number of VNFs along with their VNF descriptors (Figure 7) has been deployed in the experimental platform of the SOFTFIRE project. More specifically, three VNFs have been deployed to support OpenSAND satellite emulator (Satellite, Terminal, Gateway), three VNFs that implement open virtual switches, one VNF of a video server, one VNF of a virtual transcoder and finally one VNF of an end-user/client of the media service (Figure 8).


VNF Descriptors List of VNF Descriptors
Overview / VNF Descriptors
Add a new VNFD
Delete

Id	VNFD name	VNFD type	Vendor	Version	Actions
1616e667-361b-4b39-aa2d-8e5668e5a087	client	client	ncsrd	0.5	Action
51c04c89-f535-42b5-bfcd-25517485e9da	ovs-2	ovs-2	tub	0.2	Action
66f31994-edfc-4d98-8d57-b390bb7e33e9	ovs-1-ERIKSON	ovs-1-2intf	ncsrd	1.5	Action
74e25780-4e0c-4bf5-9bca-1b2407d5884d	video-server	video-server	ncsrd	0.5	Action
ad51babb-81d6-48ef-ab1a-68ed8bf28508	ovs-1	ovs-1	tub	0.2	Action
b08e771c-ee49-4835-84bd-f20eef77f831	satellite-emulator	satellite-emulator	ncsrd	0.6	Action
b5d7f351-d02f-47c2-b427-b86ea1cf110d	ovs-1-2intf	ovs-1-2intf	ncsrd	0.5	Action
bbc21436-dfc5-4131-b13a-28e4fe516f2c	virtual-transcoder	virtual-transcoder	tub	0.2	Action
db1a2f3d-e748-4ece-acef-b242c63b4c8f	satellite-gateway	satellite-gateway	ncsrd	0.6	Action
f5d47de3-2230-47a9-b754-087c29dce2a5	satellite-terminal	satellite-terminal	ncsrd	0.6	Action

Figure 7. Open Baton orchestrator VNF descriptors of the deployed experiment.



VNF Packages List of VNF Packages

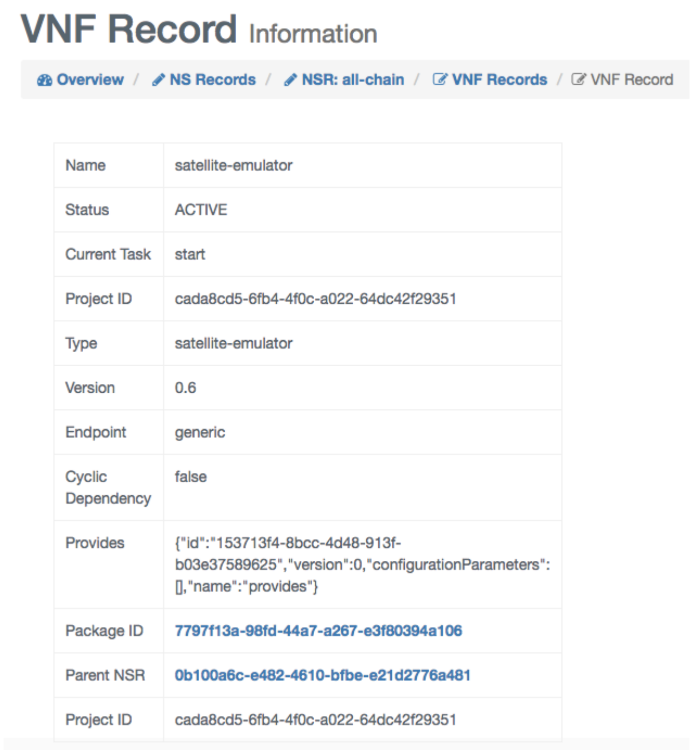
Overview / VNF Packages

	Id	Name	Version	Image	Actions
☐	4164c435-9b49-421a-a72b-1e404c874412	satellite-terminal	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	41d17bb8-f011-4aca-af0f-a94277bf3c51	satellite-gateway	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	6b5b7f21-c3a4-48a8-ab49-e8659e8c1649	OVS1	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	7797f13a-98fd-44a7-a267-e3f80394a106	satellite-emulator	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	80ad5bfe-5345-4750-81e6-4d5201ecd3d6	virtual-transcoder	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	98693225-a7cb-45d8-b1a1-3c7d50abbbe0	OVS2	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	a08407fe-74b6-4a5a-874b-96a72a293287	client	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action
☐	b507a5ff-12c6-48a3-93b2-8508d4cde78b	OVS1-2INTF	0	version: 0 , minRam: 0 , minDiskSpace: 0 , isPublic: false	Action

Figure 8. Open Baton orchestrator VNF packages of the deployed experiment.

A snapshot of the VNF descriptors list and the VNF packages as they were uploaded and deployed by the Open Baton orchestrator of the SOFTFIRE platform is depicted in the following figures.

Finally, Figure 9 depicts a representative VNF record of the OPENSAND emulator, as it was used for the execution of the EXPOSE experiments.



VNF Record Information

Overview / NS Records / NSR: all-chain / VNF Records / VNF Record

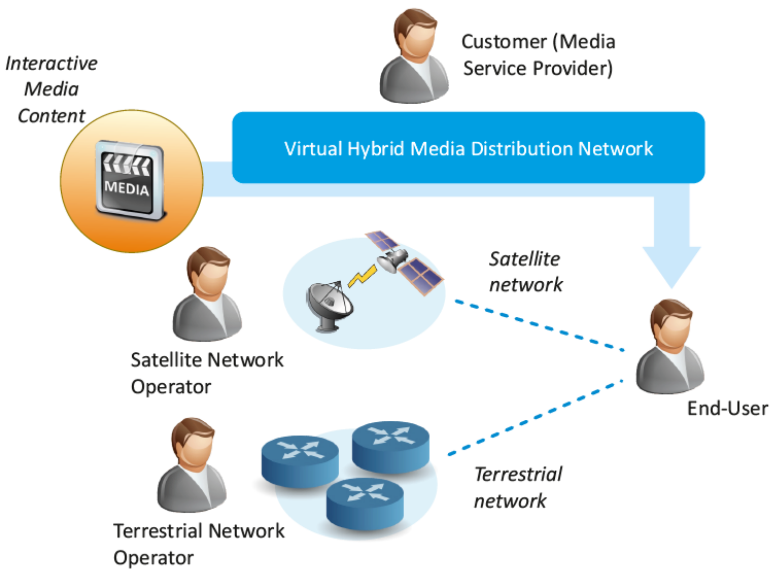
Name	satellite-emulator
Status	ACTIVE
Current Task	start
Project ID	cada8cd5-6fb4-4f0c-a022-64dc42f29351
Type	satellite-emulator
Version	0.6
Endpoint	generic
Cyclic Dependency	false
Provides	{"id":"153713f4-8bcc-4d48-913f-b03e37589625","version":0,"configurationParameters":[],"name":"provides"}
Package ID	7797f13a-98fd-44a7-a267-e3f80394a106
Parent NSR	0b100a6c-e482-4610-bfbe-e21d2776a481
Project ID	cada8cd5-6fb4-4f0c-a022-64dc42f29351

Figure 9. Open Baton orchestrator VNF record of the deployed Satellite emulator.

5.1.4.1. *Experiment #1: Hybrid media distribution network as-a-Service*

5.1.4.1.1. Experiment Description

This experiment focuses on the federation of satellite and terrestrial domains and the provision of a hybrid satellite/terrestrial access network slice to a media service provider for content distribution (see Figure 10). From a technological perspective, the scenario aims at demonstrating the agility and flexibility of SDN management over the federated infrastructure.



Interactive Media Content
MEDIA
Customer (Media Service Provider)
Virtual Hybrid Media Distribution Network
Satellite network
Satellite Network Operator
End-User
Terrestrial network
Terrestrial Network Operator

Figure 10. Hybrid media distribution network as-a-Service scenario

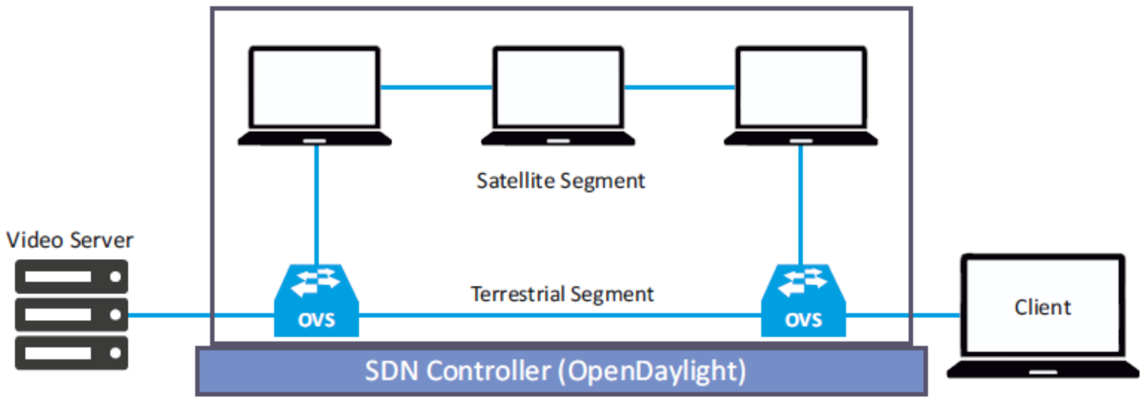


Figure 11. Hybrid media distribution network as-a-Service Experimental Topology

The experimental topology of this scenario is depicted in Figure 11, where at the ingress and egress points of the two segments (i.e. the Satellite and the Terrestrial) have been placed two SDN-compatible Open Virtual Switches (vSwitches), which are under the management and control of the OpenDaylight SDN controller. The use of SDN in this scenario permits the balancing of the load between the terrestrial and satellite segment.

In the simplest approach, the MSP (Media Service Provider/content provider) just uses the hybrid virtual network as a "dumb pipe" (yet with specific SLA) to convey media streams. However, a significant added-value of the use of virtualization and programmability technologies would be to offer to the MSP elevated management and control capabilities on the hybrid virtual network. This means that the MSP may develop his/her own network control logic in order to dynamically configure the network at runtime, allocate resources and also influence routing/forwarding decisions as desired (i.e. divert streams from the terrestrial to the satellite channel and vice versa on-the-fly or adjust the load balancing between the two networks)

Therefore, for the federation needs of this experimental scenario, an appropriate SDN-based programmability has been applied at the SDN controller in order to achieve administrative federation of the satellite and network infrastructure segment and aggregates the monitoring data of the delivered service along with the utilization statistics of the network capacity, providing immediate decisions and actions on the load balancing of the delivered traffic between the terrestrial or the satellite segment.

This experimental scenario will execute SDN-based video stream steering, aiming at presenting specific advantages of the SDN applicability on the terrestrial and satellite segment federation. Simultaneous hybrid service delivery with scalable media service is not considered in this experiment, due to the synchronization difficulties that are introduced by the different delay of the two network segments (i.e. the satellite and the terrestrial), making the implementation of the scenario beyond the scope of this report, which aims at presenting the agility and the performance efficiency of SDN.

5.1.4.2. Experiment #2: Federated terrestrial-satellite VPN

5.1.4.2.1. Experiment Description

This experiment is mostly oriented to enterprise or institutional use and assumes a customer with several distributed Points of Presence (PoPs), e.g. headquarters, branches, remote offices, mobile units etc., which need to be interconnected into an integrated corporate Virtual Private Network (VPN). We also assume that some of the PoPs are outside terrestrial network coverage, for example in isolated areas or in long-haul routes (ships, airplanes etc.). For this reason, the VPN needs to encompass both the terrestrial and the satellite infrastructures in order to cover all PoPs.

Satellite connectivity is not required only when terrestrial coverage is missing, but also in cases when a backup link is required for redundancy, when the availability requirements are strict (e.g. in mission critical applications etc.)

VPNs are commonly implemented as logically isolated overlays over the public Internet (or, less commonly, over private networks) and realized via tunneling mechanisms. All VPN endpoints have private IP addresses assigned to virtual interfaces, and appear as if they were interconnected in the same physical network.

The easiest option is to establish a VPN “over-the-top” (OTT), e.g. establish a tunneled communication with one or more remote hosts over the network, without any intervention of the network operator. This approach, although fairly simple, does not provide any service guarantees (QoS, availability etc.) and is not suitable for stable VPN networks, especially interconnecting corporate branches, which have more stringent SLA requirements.

Instead, this scenario assumes that the terrestrial and satellite operators employ virtualization and programmability technologies to offer end-to-end managed VPN services, as shown in Figure 12.

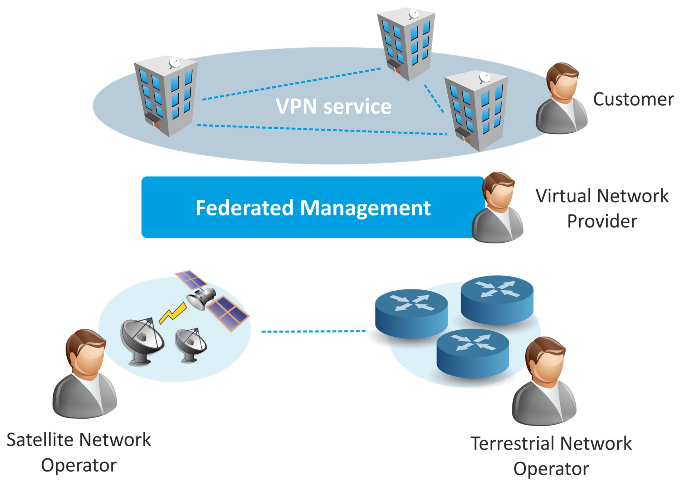


Figure 12. Federated satellite/terrestrial VPN as-a-service scenario

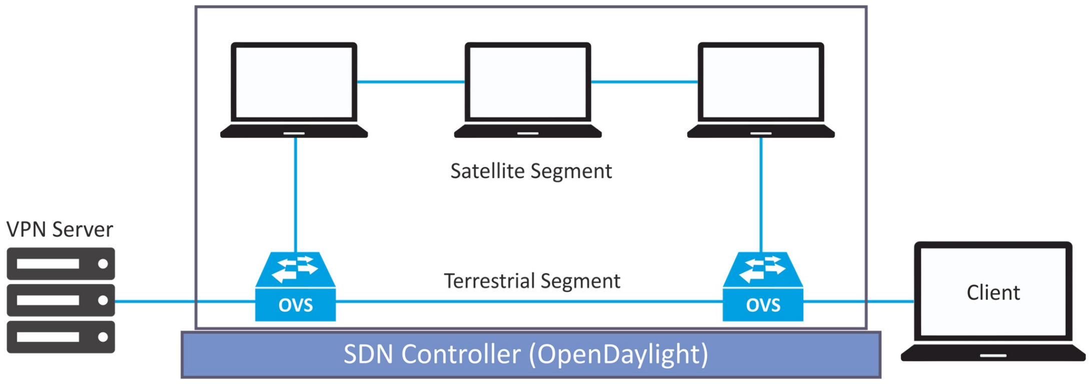


Figure 13. Federated terrestrial-satellite VPN Experimental Topology

The experimental topology of this scenario is depicted in Figure 13, where at the ingress and egress points of the two segments (i.e. the Satellite and the Terrestrial) have been placed two SDN-compatible Open Virtual Switches (vSwitches), which are under the management and control of the OpenDaylight SDN controller. The use of SDN in this scenario permits the balancing of the load between the terrestrial and satellite segment.

In this approach an openVPN server is installed at the server side of the topology, while the client side runs the respective client. Then a VPN tunneling is initiated initially over the terrestrial segment between the VPN server and the client. For the federation needs of this experimental scenario, an appropriate SDN-based programmability has been applied at the SDN controller in order to achieve administrative federation of the satellite and network infrastructure segment and steers the VPN data of the specific tunnel, providing immediate decisions and actions on the load balancing of the delivered VPN traffic between the terrestrial or the satellite segment.

This experimental scenario executes SDN-based VPN tunnel steering between the terrestrial and the satellite segment, aiming at presenting specific advantages of the SDN applicability on the terrestrial and satellite segment federation (i.e. divert VPN tunnel from the terrestrial to the satellite channel and vice versa on-the-fly or adjust the load balancing between the two networks). Simultaneous hybrid VPN delivery is not considered in this experiment, due to the synchronization difficulties that are introduced by the different delay of the two network segments (i.e. the satellite and the terrestrial), making the implementation of the scenario beyond the scope of this deliverable, which aims at presenting the agility and the performance efficiency of SDN.

5.1.4.3. Experiment #3: Dynamic backhauling with edge processing

5.1.4.3.1. Experiment Description

The satellite edge-processing scenario assumes the extension of the Mobile Edge Computing (MEC) paradigm to the satellite domain; specifically, it foresees that the backhauling service is coupled with virtualization capabilities at the satellite terminal, able to host virtual traffic processors close to the end users (Figure 14). Such local traffic processing can achieve significant savings in satellite capacity.

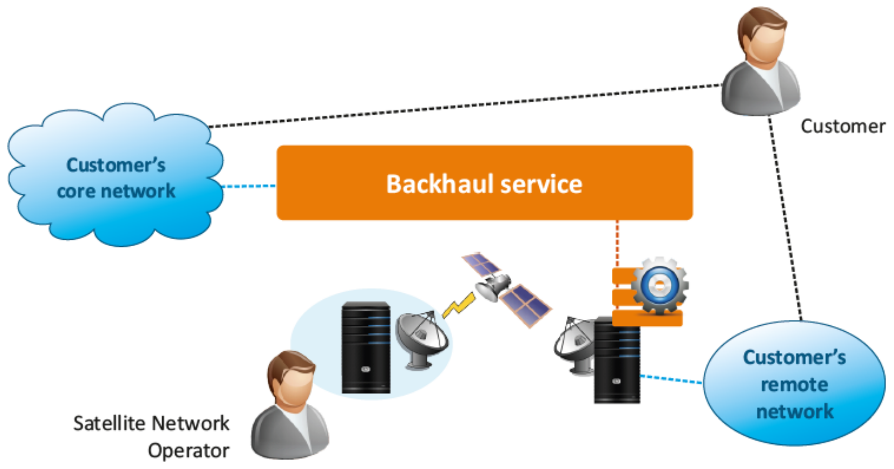


Figure 14. Dynamic backhauling with edge processing scenario

This experiment focuses on experimenting the news aggregation case, where user generated content (e.g. videos) is transmitted over the satellite towards the news aggregator server. Due to specific bandwidth availability, especially in case of multiple users, the generated video content may not be possible to be transmitted over the satellite link and therefore network congestion should result to quality degradation or even service interruption. By exploiting the Mobile Edge Computing capabilities of an appropriate VNF instantiated at the SDN/NFV-enabled Satellite Terminal, the generated streams can be dynamically transcoded and then transmitted back over satellite, minimizing the bandwidth utilization needed for the transmission of the video content and optimizing the video transmission given the total number of the video signals and the available bandwidth of the satellite link.

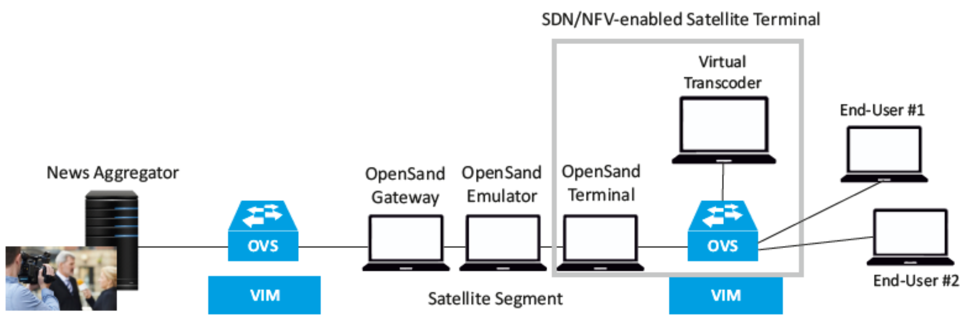


Figure 15. Dynamic backhauling with edge processing Experimental Topology

The experimental topology of this scenario is depicted in Figure 15, where at the ingress and egress points of the satellite segments they have been placed two SDN-compatible Open Virtual Switches, which are under the management and control of the OpenDaylight SDN controller.

With regard to edge processing, the NFV coupled with emerging Mobile Edge Computing (MEC) concepts for deployment of cloud resources at the network edge, are the key enabling technologies. The satellite terminal needs to encompass virtualized IT resources in order to host the traffic processors, as virtual network functions (VNFs). Thus, at the edge of the satellite segment, it is considered an SDN/NFV-enabled Satellite Terminal, which is capable of being controlled by the OpenBaton Orchestrator, instantiating appropriate VNFs and implementing upon the SDN controller mandates the required SDN rules for performing traffic steering as needed for the Service Function Chaining (SFC).

This experiment (Live news gathering with dynamic transcoding) aims at presenting specific advantages of the SDN/NFV applicability at the edge of the satellite segment, such as real-time service adaptation, minimization of the satellite link utilization, achieving scalability in case of multiple end-users by applying network resource elasticity per end-user.

5.1.4.4. KPIs and performance metrics

The following KPIs (reported in below tables) are used as means of verification of the executed experiments and are presented in details in the next section 5.1.5 of this deliverable.

KPI-1: Deployment and integration in SoftFIRE of OpenSAND Satellite emulator as a VNF
Description
OpenSAND Satellite emulator deployed by jFed tool/FITEagle/OpenBaton is up and running and integrated with the SoftFIRE infrastructure.
Type
Boolean (**passed** / not passed)
Means of verification
This KPI will be verified by instantiating the three VMs of OpenSAND emulator and two VMs as users in SoftFIRE OpenStack infrastructure domain through the jFed tool. jFed will then communicate with the FITEagle instance running in the SoftFIRE testbed. FITEagle is preconfigured in order to be able to interface with the northbound API of an Open Baton instance. Finally Open Baton will be in charge of deploying the requested NFV experiments in the SoftFIRE platform. This setup guarantees that the integration of satellite emulator has been completed.
Status: **ACHIEVED**

Table 2: Expose Table of KPI1

KPI-2: Steering the traffic between the satellite emulator and the terrestrial domain
Description
This KPI focuses on the federation of satellite and terrestrial domains and the provision of a hybrid satellite/terrestrial access network for service distribution. The setup facilitates the service delivery under different operating conditions, showing that this approach can achieve better performance than a single domain network.

Type Quantitative: Measurement of performance related metrics (e.g. throughput, latency, etc.). **Means of verification** This KPI will be verified by instantiating a hybrid network topology and the traffic to be steered between the satellite and the terrestrial domain. The network performance metrics (e.g. throughput, latency etc.) from the ingress point of the hybrid network segment to the egress point (i.e. the end-user) will be measured under different operating conditions (e.g. background traffic). Status: **ACHIEVED**

Table 3: Expose Table of KPI2

KPI-3: Transcoding of the delivered video service over the hybrid experimental testbed **Description** This KPI focuses on facilitating the video transmission in a dynamic and transparent way for the end-users over a satellite domain by following a MEC strategy and instantiating a transcoder near to the terminal-side of the OpenSAND satellite emulator. This setup reduces the utilization of the costly satellite link and facilitates the service delivery under different operating conditions, showing that better performance can be achieved. **Type** Quantitative: Measurement of video-related metrics (e.g. video quality) **Means of verification** This KPI will be verified at the hybrid experimental testbed by instantiating a transcoder as VNF at the terminal-side of the testbed and applying appropriate traffic steering rules to transparently steer the media flow through the VM-based transcoder and then the transcoded service to be forwarded over the satellite in order to finally reach the Gateway-side. It will be shown that the proposed KPI can contribute towards service continuation of the video through the proposed metrics (e.g. video quality, frame rate) under different operating conditions. Status: **ACHIEVED**

Table 4: Expose Table of KPI3

5.1.5. Results analysis and KPIs interpretation

5.1.5.1. EXPERIMENT #1 : Hybrid media distribution network as-a-Service

5.1.5.1.1. Experiment Execution

This experiment presents the SDN-based video stream steering between the satellite and the terrestrial segments, considering that a unicast video service is initially delivered over the terrestrial network at the pre-defined QoE level and then due to service degradation (or other triggering event) the service delivery is switched seamlessly via the satellite segment.

Thus, in this experiment, the unicast media streams are load-balanced between the satellite and the terrestrial segment, according to the available network resources. We assume that the primary distribution channel should be the terrestrial one; the customer receives the media content over terrestrial and when insufficient terrestrial capacity is observed, the traffic is diverted by appropriate SDN multipath rules over the satellite segment.

The experiment can be also executed in the reverse order, where the primary delivery channel is different and considers the switching of the video service from the satellite segment to the terrestrial segment. The storyline of this scenario is the following:

1. Video service delivered over terrestrial network
2. Background terrestrial network traffic degrades video quality
3. Federator monitors and applies appropriate traffic steering SDN rule
4. SDN-rule is applied (L2 forwarding over satellite)
5. Video quality is reinstated

Figure 16 depicts the initiation of the unicast video service from the content server (192.168.20.26) over the terrestrial link towards the end-user 192.168.21.27 at port 33334 utilizing the MPEG-4 video codec using Simple Profile with spatial resolution 640x480, frame rate 24 fps and bitrate ~1024 kbps.

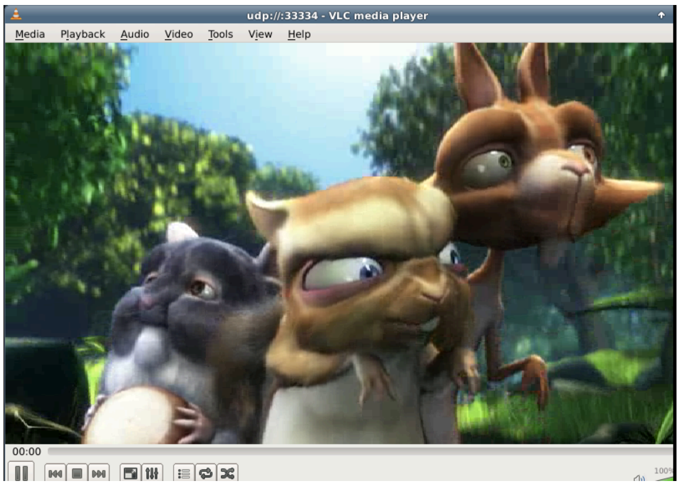
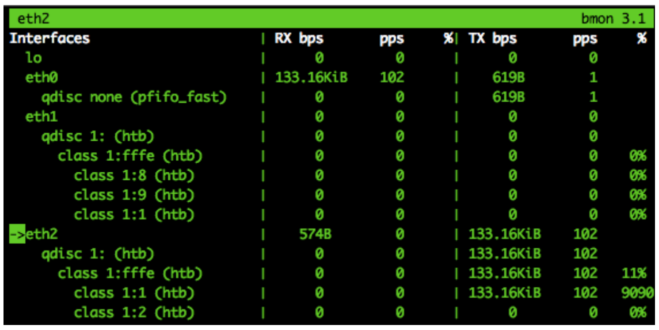


Figure 16. Normal media delivery over terrestrial

The unicast video traffic successfully passes through the entry switch (ovs-1), where it is monitored by the Federator, as the following figure depicts, measuring a flow rate of 133.16 kB (i.e. approx. 1Mbps) at eth2, which is the port that leads to the terrestrial segment. During the delivery of the unicast media service the terrestrial network link, as depicted on Figure 17, utilizes approximately 11% of the overall available network bandwidth.



```
 eth2                                                          bmon 3.1
Interfaces                   | RX bps       pps     %| TX bps       pps     %
  lo                         |      0         0      |      0         0
  eth0                       | 133.16KiB    102      |    619B        1
    qdisc none (pfifo_fast)  |      0         0      |    619B        1
  eth1                       |      0         0      |      0         0
    qdisc 1: (htb)           |      0         0      |      0         0
      class 1:fffe (htb)     |      0         0      |      0         0    0%
        class 1:8 (htb)      |      0         0      |      0         0    0%
        class 1:9 (htb)      |      0         0      |      0         0    0%
        class 1:1 (htb)      |      0         0      |      0         0    0%
->eth2                       |    574B        0      | 133.16KiB    102
    qdisc 1: (htb)           |      0         0      | 133.16KiB    102
      class 1:fffe (htb)     |      0         0      | 133.16KiB    102   11%
        class 1:1 (htb)      |      0         0      | 133.16KiB    102  9090
        class 1:2 (htb)      |      0         0      |      0         0    0%
```

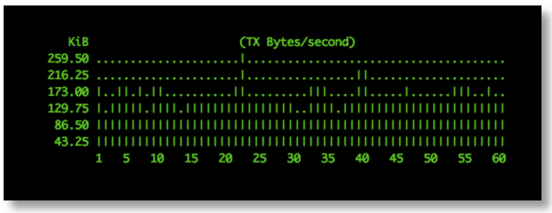


Figure 17. Media Delivery over terrestrial

The unicast media stream is delivered over the terrestrial link, it can be seen that the ICMP RTT to the video server is below 1 msec (Figure 18).

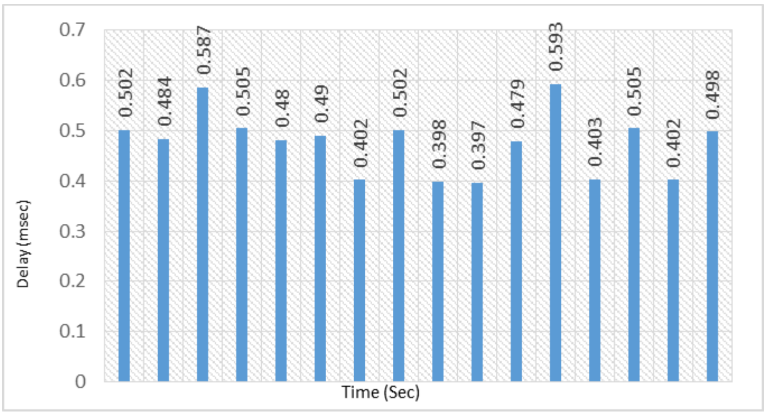
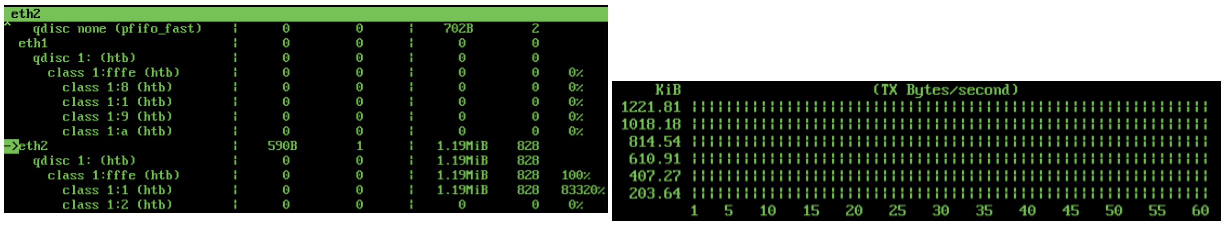


Figure 18. RTT to the video server over the terrestrial link

According to the story line, background traffic is added in the terrestrial channel in order to force quality degradation of the delivered video service. For the experimental needs of the scenario, the maximum available bandwidth of each interface has been reduced to 10 Mbit in order to be facilitated to traffic flooding of the channel with background traffic.

Towards flooding the satellite link with background traffic, synthetic UDP traffic is generated by a Linux virtual machine utilizing the iperf command similarly to the previous experiment. The produced traffic creates approximately traffic of 10Mbit, which is enough in order to flood the link with 70% utilization and therefore creating quality degradation to the delivered video service due to network impairments, such as jitter, delay etc. This congested network link is depicted in Figure 19, where traffic of 1.19 MB/sec is monitored at eth2 and the quality degradation at the SSIM metric is depicted on Figure 20



```
eth2
^  qdisc none (pfifo_fast)    ¦     0          0     ¦    702B      2
 eth1                         ¦     0          0     ¦     0        0
   qdisc 1: (htb)             ¦     0          0     ¦     0        0
     class 1:fffe (htb)       ¦     0          0     ¦     0        0    0%
       class 1:8 (htb)        ¦     0          0     ¦     0        0    0%
       class 1:1 (htb)        ¦     0          0     ¦     0        0    0%
       class 1:9 (htb)        ¦     0          0     ¦     0        0    0%
       class 1:a (htb)        ¦     0          0     ¦     0        0    0%
->eth2                        ¦   590B         1     ¦  1.19MiB   828
   qdisc 1: (htb)             ¦     0          0     ¦  1.19MiB   828
     class 1:fffe (htb)       ¦     0          0     ¦  1.19MiB   828   100%
       class 1:1 (htb)        ¦     0          0     ¦  1.19MiB   828 83320%
       class 1:2 (htb)        ¦     0          0     ¦     0        0    0%
```

```
     KiB                         (TX Bytes/second)
1221.81 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
1018.18 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
 814.54 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
 610.91 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
 407.27 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
 203.64 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
        1   5    10   15   20   25   30   35   40   45   50   55   60
```

Figure 19. Background traffic introduced in terrestrial link

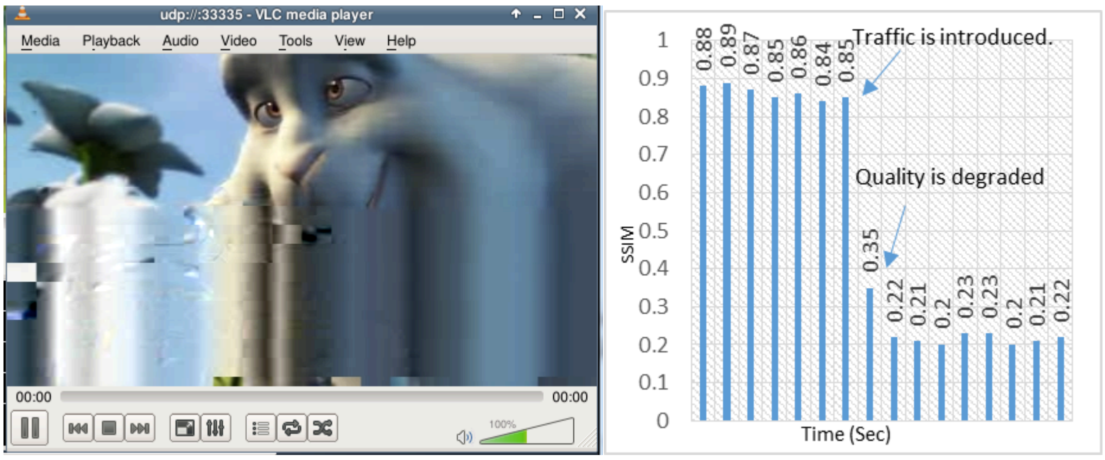
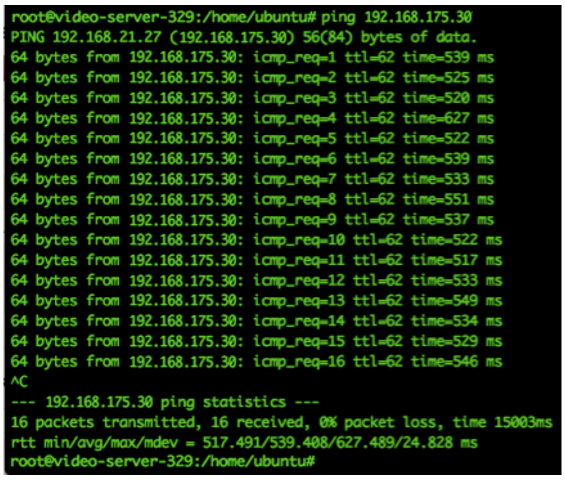


Figure 20. Severe video quality degradation

Upon monitoring the quality degradation, appropriate SDN-based traffic steering commands at the ingress OVS of the experimental topology (i.e. ovs-1) divert the media to the satellite link. Figure 21 depicts both the satellite domain port and the terrestrial domain port, where it is observed that the media service (approx. 144.93kB/sec) is delivered over the satellite link (through eth1 port) and the rest background traffic (approx. 732.34kB/sec) continues to be delivered over the terrestrial link (through eth2). The delivery of the media service over the satellite link is confirmed also by the measurement of the one way day between the MSP and the client, which has been increased at approx. 535 msec.



```
root@video-server-329:/home/ubuntu# ping 192.168.175.30
PING 192.168.21.27 (192.168.175.30) 56(84) bytes of data.
64 bytes from 192.168.175.30: icmp_req=1 ttl=62 time=539 ms
64 bytes from 192.168.175.30: icmp_req=2 ttl=62 time=525 ms
64 bytes from 192.168.175.30: icmp_req=3 ttl=62 time=520 ms
64 bytes from 192.168.175.30: icmp_req=4 ttl=62 time=627 ms
64 bytes from 192.168.175.30: icmp_req=5 ttl=62 time=522 ms
64 bytes from 192.168.175.30: icmp_req=6 ttl=62 time=539 ms
64 bytes from 192.168.175.30: icmp_req=7 ttl=62 time=533 ms
64 bytes from 192.168.175.30: icmp_req=8 ttl=62 time=551 ms
64 bytes from 192.168.175.30: icmp_req=9 ttl=62 time=537 ms
64 bytes from 192.168.175.30: icmp_req=10 ttl=62 time=522 ms
64 bytes from 192.168.175.30: icmp_req=11 ttl=62 time=517 ms
64 bytes from 192.168.175.30: icmp_req=12 ttl=62 time=533 ms
64 bytes from 192.168.175.30: icmp_req=13 ttl=62 time=549 ms
64 bytes from 192.168.175.30: icmp_req=14 ttl=62 time=534 ms
64 bytes from 192.168.175.30: icmp_req=15 ttl=62 time=529 ms
64 bytes from 192.168.175.30: icmp_req=16 ttl=62 time=546 ms
^C
--- 192.168.175.30 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15003ms
rtt min/avg/max/mdev = 517.491/539.408/627.489/24.828 ms
root@video-server-329:/home/ubuntu#
```

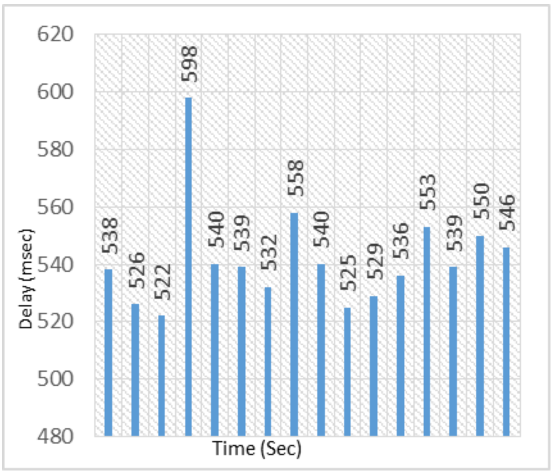
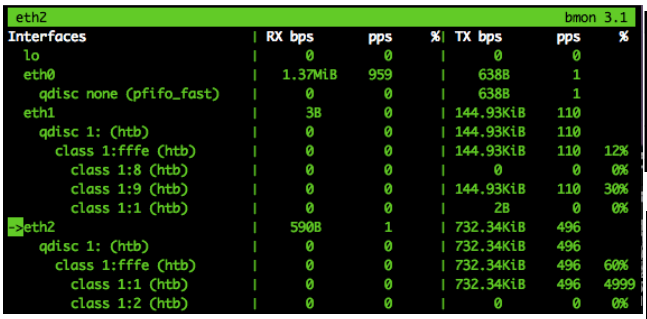
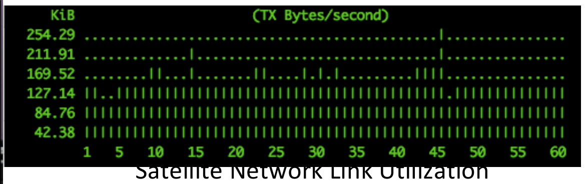


Figure 21. ICMP RTT to the video server over the satellite link

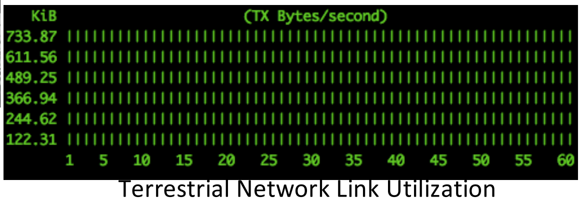
In this traffic steering experiment, the network utilization of the satellite link is approx. 10%, while the network utilization of the terrestrial link is approx. 60%, as Figure 22 depicts.



```
 eth2                                                              bmon 3.1
Interfaces                     | RX bps       pps     %| TX bps       pps     %
  lo                           |      0         0      |      0          0
  eth0                         |  1.37MiB     959      |    638B         1
    qdisc none (pfifo_fast)    |      0         0      |    638B         1
  eth1                         |     3B         0      | 144.93KiB     110
    qdisc 1: (htb)             |      0         0      | 144.93KiB     110
      class 1:fffe (htb)       |      0         0      | 144.93KiB     110   12%
        class 1:8 (htb)        |      0         0      |      0          0    0%
        class 1:9 (htb)        |      0         0      | 144.93KiB     110   30%
        class 1:1 (htb)        |      0         0      |     2B          0    0%
->eth2                         |   590B         1      | 732.34KiB     496
    qdisc 1: (htb)             |      0         0      | 732.34KiB     496
      class 1:fffe (htb)       |      0         0      | 732.34KiB     496   60%
        class 1:1 (htb)        |      0         0      | 732.34KiB     496  4999
        class 1:2 (htb)        |      0         0      |      0          0    0%
```



Satellite Network Link Utilization



Terrestrial Network Link Utilization

Figure 22. Media service is switced over Satellite, while background traffic remains at the terrestrial

Upon the traffic steering of the media service over the satellite link, the video quality of the media service (SSIM) is reinstated at approximately 0.85 from approx. 0.21 as it is depicted on Figure 23

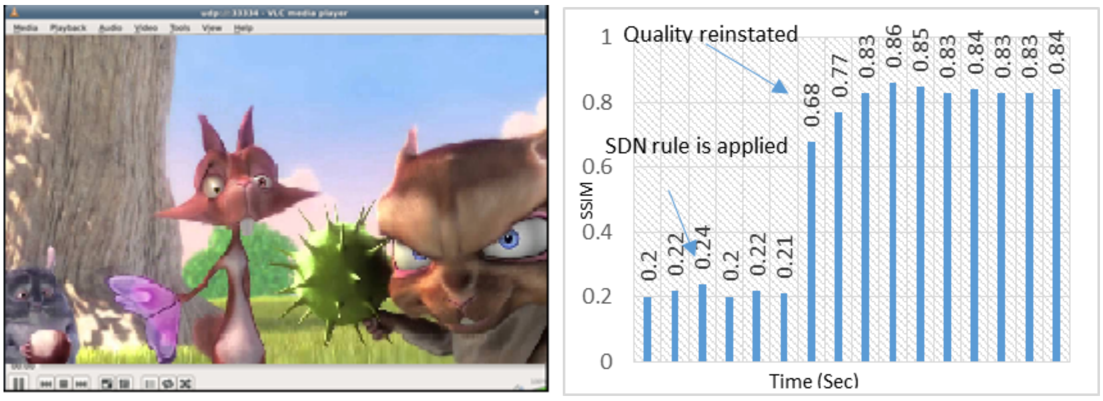


Figure 23. Video Quality is reinstated

5.1.5.1.2. Conclusions and Added-value

Contemporary virtualization technologies allow network operators to partition their networks into virtual slices, with specific capacity and QoS, and to offer these slices to content service providers. This capability is promoted more and more via EU and global research efforts, as well as novel network management architectures and even close-to-market products.

This experiment extends this concept to also embrace the satellite segment. Virtualisation technologies can abstract the satellite and terrestrial access network and also federate them, so they can be offered to the Media Service Provider as a single logically isolated virtual infrastructure, as-a-Service. In this manner, the media provider can extend his/her customer coverage area, almost without any requirement for upfront investment.

A significant added-value of this experiment is that the media service provider has elevated management and control capabilities on the hybrid virtual network. This means that the media provider may develop his/her own network control logic in order to dynamically configure the network at runtime, allocate resources and also influence routing/forwarding decisions as desired (i.e. divert streams from the terrestrial to the satellite channel and vice versa on-the-fly or adjust the load balancing between the two networks)

Furthermore, thanks to resource elasticity, the capacity and QoS offered to the MSP virtual network may fluctuate over time, enabling the MSP service to be up and down scaled on-demand or automatically, to react to the customers' demand. This means that the MSP may dynamically request more capacity if needed (e.g. in case of highly popular content)

5.1.5.2. Experiment #2: Federated terrestrial-satellite VPN

5.1.5.2.1. Experiment Execution

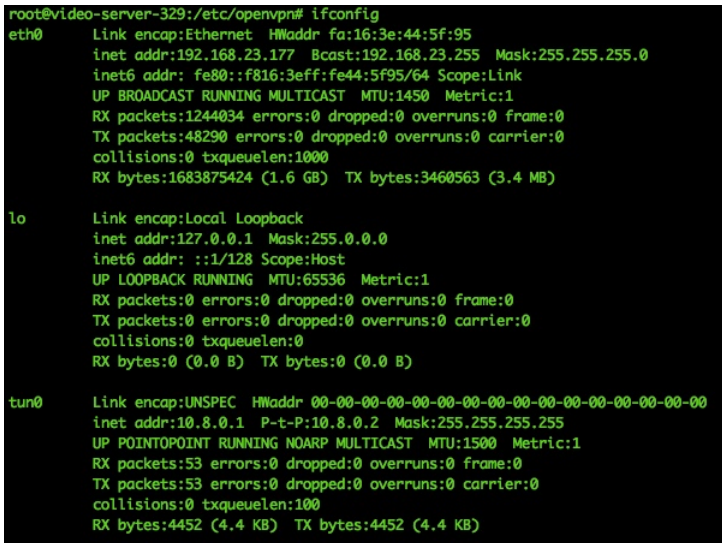
This experiment presents the SDN-based VPN tunneling steering between the satellite and the terrestrial segments, considering that a unique VPN tunnel is initially established over the terrestrial network at the pre-defined QoE level and then due to a triggering event the VPN tunnel is switched seamlessly via the satellite segment (based on SDN steering).

Thus, in this experiment, the unicast media streams are load-balanced between the satellite and the terrestrial segment, according to the available network resources. We assume that the primary distribution channel should be the terrestrial one, and then the VPN tunnel is diverted by appropriate SDN multipath rules over the satellite segment.

The experiment can be also executed in the reverse order, where the primary VPN tunnel is different and considers the switching of the VPN tunnel from the satellite segment to the terrestrial segment. The storyline of this scenario is the following:

1. VPN tunnel established over terrestrial network
2. Triggering event occurs
3. SDN-rule is applied (L2 forwarding over satellite)
4. VPN tunnel is steered over the satellite link

Figure 24 depicts the establishment of the VPN tunnel at the server side, while Figure 25 depicts the other end of the tunnel at the client side.

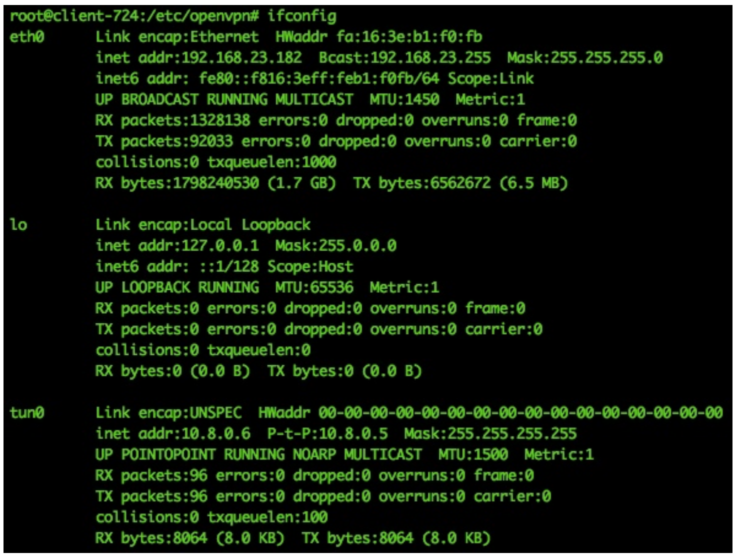


```
root@video-server-329:/etc/openvpn# ifconfig
eth0      Link encap:Ethernet  HWaddr fa:16:3e:44:5f:95
          inet addr:192.168.23.177  Bcast:192.168.23.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe44:5f95/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:1244034 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1683875424 (1.6 GB)  TX bytes:3460563 (3.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1  P-t-P:10.8.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:53 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:4452 (4.4 KB)  TX bytes:4452 (4.4 KB)
```

Figure 24. VPN tunnel interface at the server side



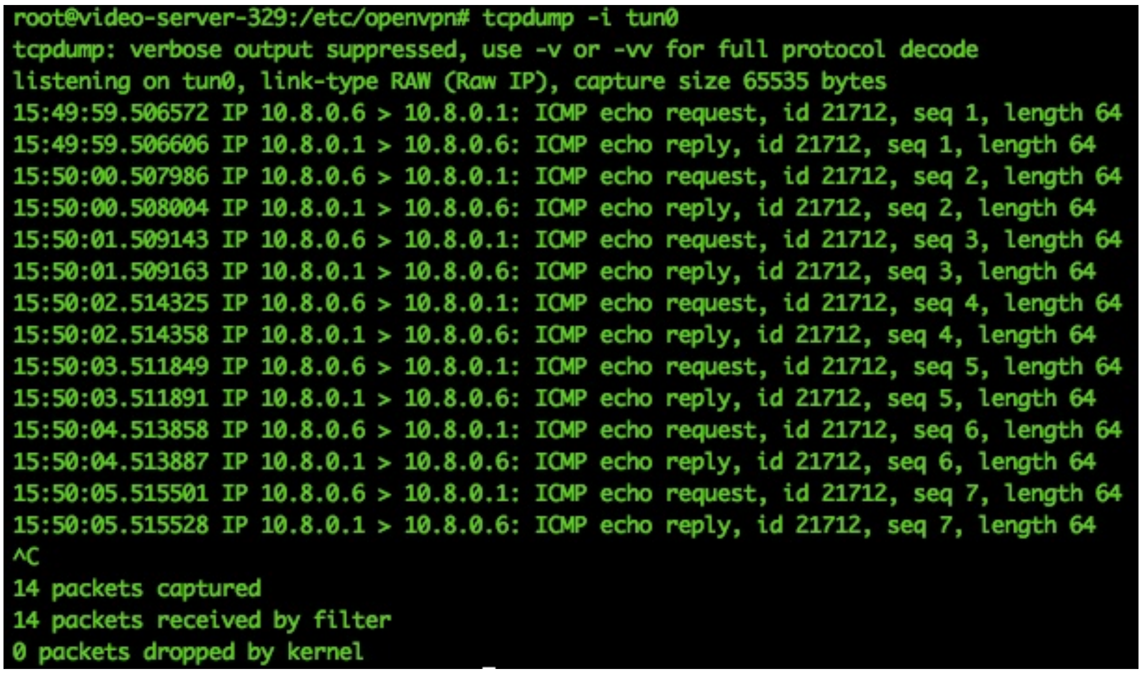
```
root@client-724:/etc/openvpn# ifconfig
eth0      Link encap:Ethernet  HWaddr fa:16:3e:b1:f0:fb
          inet addr:192.168.23.182  Bcast:192.168.23.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:feb1:f0fb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:1328138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92033 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1798240530 (1.7 GB)  TX bytes:6562672 (6.5 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.6  P-t-P:10.8.0.5  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:8064 (8.0 KB)  TX bytes:8064 (8.0 KB)
```

Figure 25. VPN tunnel interface at the client side

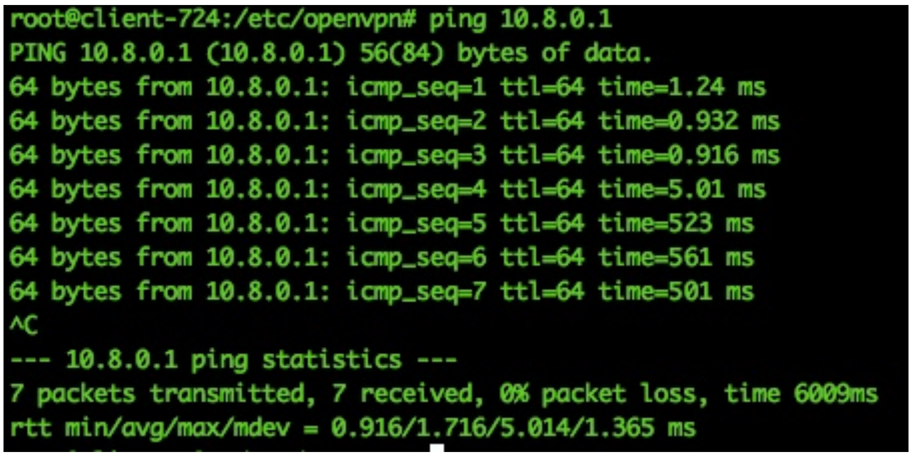
For validating the deployment of the VPN tunnel a tcp-dump was performed, which captured the VPN traffic and a representative snapshot is presented in Figure 26.



```
root@video-server-329:/etc/openvpn# tcpdump -i tun0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 65535 bytes
15:49:59.506572 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 1, length 64
15:49:59.506606 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 1, length 64
15:50:00.507986 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 2, length 64
15:50:00.508004 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 2, length 64
15:50:01.509143 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 3, length 64
15:50:01.509163 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 3, length 64
15:50:02.514325 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 4, length 64
15:50:02.514358 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 4, length 64
15:50:03.511849 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 5, length 64
15:50:03.511891 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 5, length 64
15:50:04.513858 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 6, length 64
15:50:04.513887 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 6, length 64
15:50:05.515501 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 21712, seq 7, length 64
15:50:05.515528 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 21712, seq 7, length 64
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
```

Figure 26. TCPdump of the VPN tunnel

The experiment was executed based on SDN-based VPN tunnel steering between the terrestrial and the satellite segment, aiming at presenting specific advantages of the SDN applicability on the terrestrial and satellite segment federation (i.e. divert VPN tunnel from the terrestrial to the satellite channel and vice versa on-the-fly or adjust the load balancing between the two networks). Figure 27 depicts this on-the-fly transition, which is observed by the different in the ping delay time.



```
root@client-724:/etc/openvpn# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.932 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.916 ms
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=5.01 ms
64 bytes from 10.8.0.1: icmp_seq=5 ttl=64 time=523 ms
64 bytes from 10.8.0.1: icmp_seq=6 ttl=64 time=561 ms
64 bytes from 10.8.0.1: icmp_seq=7 ttl=64 time=501 ms
^C
--- 10.8.0.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6009ms
rtt min/avg/max/mdev = 0.916/1.716/5.014/1.365 ms
```

Figure 27. SDN-based VPN tunnel steering between the terrestrial and the satellite segment

Simultaneous hybrid VPN delivery is not considered in this experiment, due to the synchronization difficulties that are introduced by the different delay of the two network segments (i.e. the satellite and the terrestrial), making the implementation of the scenario beyond the scope of this report, which aims at presenting the agility and the performance efficiency of SDN. Moreover, should be reported that the system during the steering process of the VPN tunnelling between the two segments was not sufficiently stable, resulting in re-occurring disconnections of the VPN tunnelling, which didn't allow us to proceed with further experimentation and measurements.

5.1.5.2.2. Conclusions and Added-value

The use of programmability and virtualization technologies for the establishment and control of the VPN would bring several valuable benefits such as:

- Rapid setup as well as reconfiguration of the VPN service, with a delay of minutes or even seconds. This capability is especially useful in cases where the service needs to be quickly deployed and/or reconfigured i.e. disaster recovery or high mobility
- Unified control of the satellite and terrestrial domains via standardized protocols, enabling vendor-agnostic setup of VPNs flexibly using also plain IP encapsulation.
- Direct mesh routing without the need of a VPN concentrator. Via centralized control, traffic can be encapsulated close to the user and re-routed through the network directly to the peer. This means that the typical “hub and spoke” VPN topology can be avoided, allowing the traffic to be directly diverted to the peer node, increasing network efficiency and minimizing latency.
- Better support of user mobility i.e. dynamic reconfiguration of the VPN network as one of the end nodes roams across networks and its attachment address changes
- More efficient monitoring of the entire VPN service, providing detailed insight of the traffic in all branches of the VPN topology.
- Elastic resource scaling, on-demand allocation and flexible billing
- Exposure of advanced control capabilities to the customer over the VPN. That is, the customer may (in some cases) be offered the capability to apply some arbitrary flow handling logic over the VPN e.g. block/prioritise/reroute flows etc. This is in line with the “SDN-as-a-Service” paradigm.

5.1.5.3. Experiment #3: Dynamic backhauling with edge processing

5.1.5.3.1. Experiment Execution

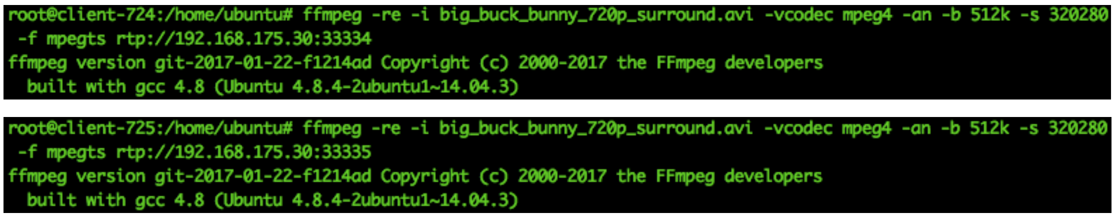
This experiment considers that two end users wish to simultaneously transmit user-generated video content back to the video news aggregator via the same (professional) satellite terminal. Therefore two discrete unicast flows are initiated by the end-users with final destination the remote news aggregator server. However, due to limited satellite bandwidth the two media streams exceed the available bandwidth in the satellite link resulting in network congestion and therefore degradation of the QoE of the transmitted media signals.

Towards facilitating the video transmission in a dynamic and transparent way for the end-users, the experimenter monitors the quality degradation and by following a MEC strategy instantiates a transcoder as VNF at the SDN/NFV-enabled terminal and applies appropriate SDN-based traffic steering rules at the OVSs to transparently steer the two media flows through the VNF-based transcoder and then to be forwarded over the satellite in order to finally reach the news aggregator. The storyline of this scenario is the following:

1. Two end-users are sending (each one) a unicast media service to the news aggregator
2. Both media services are transmitted over the satellite network uplink
3. Total traffic of the two media services exceeds the available bandwidth of the satellite link resulting to network congestion and quality degradation.

4. The experimenter monitors and instantiates at the SDN/NFV-enabled satellite terminal a VNF-based transcoder and appropriate SDN rules for the traffic steering in order to support the SFC.
5. Both media streams are transcoded (i.e. at lower bitrate/frame rate/resolution) in real time and transparently from the end-users
6. The two transcoded media services can fit in the available satellite bandwidth and are transmitted without any network congestion.
7. The QoE level of the two transmitted signals is re-instated at satisfactory levels and the two signals reach the remote news aggregator server without impairments.

Initially each user initiates a unicast video from its terminal back to the news aggregator over the satellite return link. Figure 28 depicts this initiation from each one of the two terminals. The first terminal transmits its media service at the port 33334 and the second terminal at the port 33335.



```
root@client-724:/home/ubuntu# ffmpeg -re -i big_buck_bunny_720p_surround.avi -vcodec mpeg4 -an -b 512k -s 320280
 -f mpegts rtp://192.168.175.30:33334
ffmpeg version git-2017-01-22-f1214ad Copyright (c) 2000-2017 the FFmpeg developers
  built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
```

```
root@client-725:/home/ubuntu# ffmpeg -re -i big_buck_bunny_720p_surround.avi -vcodec mpeg4 -an -b 512k -s 320280
 -f mpegts rtp://192.168.175.30:33335
ffmpeg version git-2017-01-22-f1214ad Copyright (c) 2000-2017 the FFmpeg developers
  built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
```

Figure 28. Initiation of video flows

The bitrate of each video is approximately 256 kbps, so the two streams sub up at approximately 512 kbps, which exceeds the capacity of the emulated DVB-RCS return channel. The allocated return channel satellite terminal (RCST) capacity is constrained to approx. 500 kbps. In our experimental case the total uploading of the two individual media services (i.e. 512 kbps each) exceeds the available uplink capacity resulting to severe quality degradation of both media services at the media aggregator side as it is depicted on Figure 29.

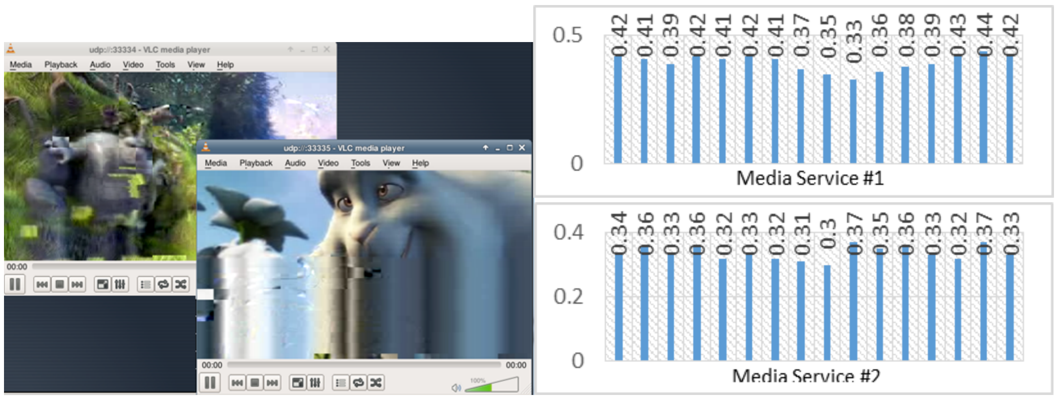
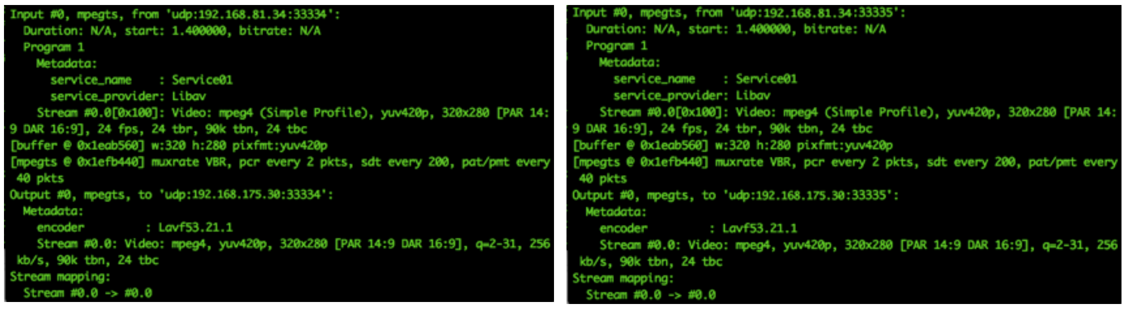


Figure 29. Degradation of the two streams caused by exceeding the capacity of satellite link

Towards dynamically and seamlessly improving the media service delivery, the experimenter instantiates the transcoder VNF at the SDN/NFV-enabled satellite terminal, as it is depicted in Figure 30, and performs in real time and totally seamlessly to the end-user either spatial or temporal or transcoding (or combination of them) in both video signals. It should be pointed out that for each media service a different instance of the transcoder VNF is instantiated, which results on two different instances (one for the media service at port 33334 and one for the service at the port 33335) that are executed on the SDN/NFV-enabled satellite terminal.



```
Input #0, mpegts, from 'udp:192.168.81.34:33334':
  Duration: N/A, start: 1.400000, bitrate: N/A
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: Libav
    Stream #0.0[0x100]: Video: mpeg4 (Simple Profile), yuv420p, 320x280 [PAR 14:
9 DAR 16:9], 24 fps, 24 tbr, 90k tbn, 24 tbc
[buffer @ 0x1eab560] w:320 h:280 pixfmt:yuv420p
[mpegts @ 0x1efb440] muxrate VBR, pcr every 2 pkts, sdt every 200, pat/pmt every
 40 pkts
Output #0, mpegts, to 'udp:192.168.175.30:33334':
  Metadata:
    encoder         : Lavf53.21.1
    Stream #0.0: Video: mpeg4, yuv420p, 320x280 [PAR 14:9 DAR 16:9], q=2-31, 256
 kb/s, 90k tbn, 24 tbc
Stream mapping:
  Stream #0.0 -> #0.0
```

```
Input #0, mpegts, from 'udp:192.168.81.34:33335':
  Duration: N/A, start: 1.400000, bitrate: N/A
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: Libav
    Stream #0.0[0x100]: Video: mpeg4 (Simple Profile), yuv420p, 320x280 [PAR 14:
9 DAR 16:9], 24 fps, 24 tbr, 90k tbn, 24 tbc
[buffer @ 0x1eab560] w:320 h:280 pixfmt:yuv420p
[mpegts @ 0x1efb440] muxrate VBR, pcr every 2 pkts, sdt every 200, pat/pmt every
 40 pkts
Output #0, mpegts, to 'udp:192.168.175.30:33335':
  Metadata:
    encoder         : Lavf53.21.1
    Stream #0.0: Video: mpeg4, yuv420p, 320x280 [PAR 14:9 DAR 16:9], q=2-31, 256
 kb/s, 90k tbn, 24 tbc
Stream mapping:
  Stream #0.0 -> #0.0
```

Figure 30. Initiation and operation of the two transcoder VNF instances

Figure 31depicts the total traffic before and after the transcoding process, as monitored at the NIC of the VM, on which the two instances of the transcoder VNF have been instantiated.

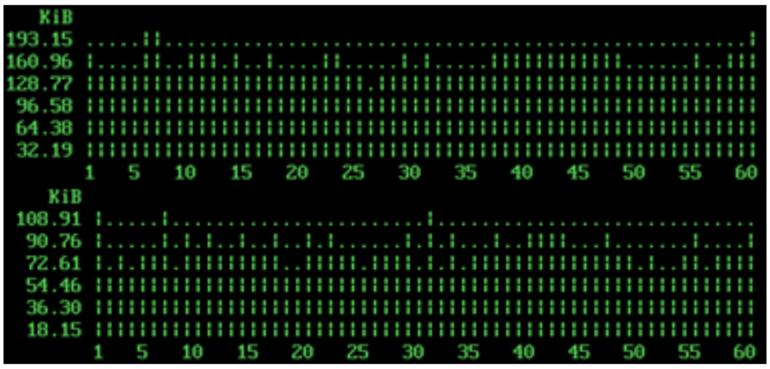
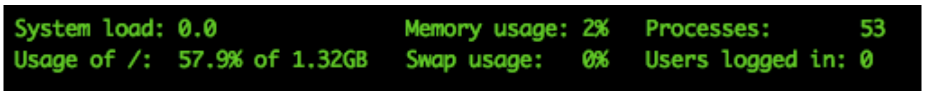


Figure 31. Total Media Service Traffic Before and After Transcoding

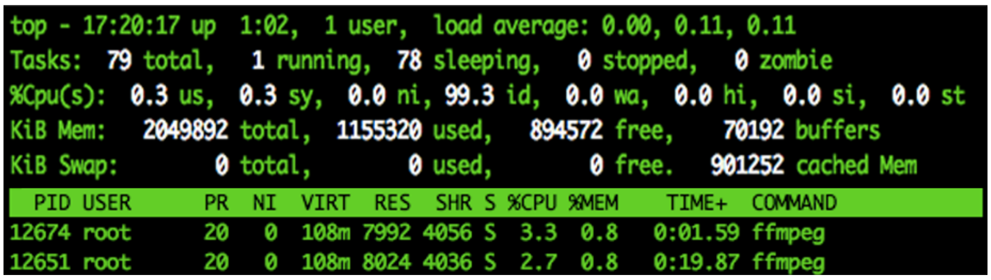
In terms of VNF performance evaluation, the VM that hosts the two instances of the transcoder VNF has been assigned 2GB of RAM, 23GB of HDD size and 1 CPU core (2.4GHz). As it is depicted on Figure 32, during VNF operation, the VM has an overall system load of 5% in terms of CPU utilization, 9.4% HDD utilization and 36% memory usage, allowing the VM to operate on a stable and efficient status.



```
System load: 0.0            Memory usage: 2%    Processes:         53
Usage of /:  57.9% of 1.32GB  Swap usage:  0%    Users logged in: 0
```

Figure 32. Performance parameters (%CPU, %MEM, %HDD) of the VM hosting the VNFs

Considering the performance of each instance of the VNF running at the specific VM, Figure 33 depicts that each instance utilizes approximately 3% of the CPU (instance #1 3.3%, instance #2 2.7%), while each of them occupies only 0.8% of the available RAM. Thus, for the selected configuration, the two VNFs consume a relatively low amount of resources. Of course, this amount is only indicative and depends on the implementation of the transcoder and also on the bitrate and format of the streams being transcoded.



```
top - 17:20:17 up  1:02,  1 user,  load average: 0.00, 0.11, 0.11
Tasks:  79 total,   1 running,  78 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2049892 total,  1155320 used,   894572 free,    70192 buffers
KiB Swap:        0 total,        0 used,        0 free.   901252 cached Mem
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+ COMMAND
12674 root      20   0  108m 7992 4056 S  3.3  0.8   0:01.59 ffmpeg
12651 root      20   0  108m 8024 4036 S  2.7  0.8   0:19.87 ffmpeg
```

Figure 33. Performance parameters of the two VNF instances (%CPU, %MEM)

Upon the real time transcoding of the two media services from 512kbps down to 256 kbps each, the video quality is reinstated seamlessly (i.e. without requiring any interruption) for both signals as it is depicted in Figure 34.

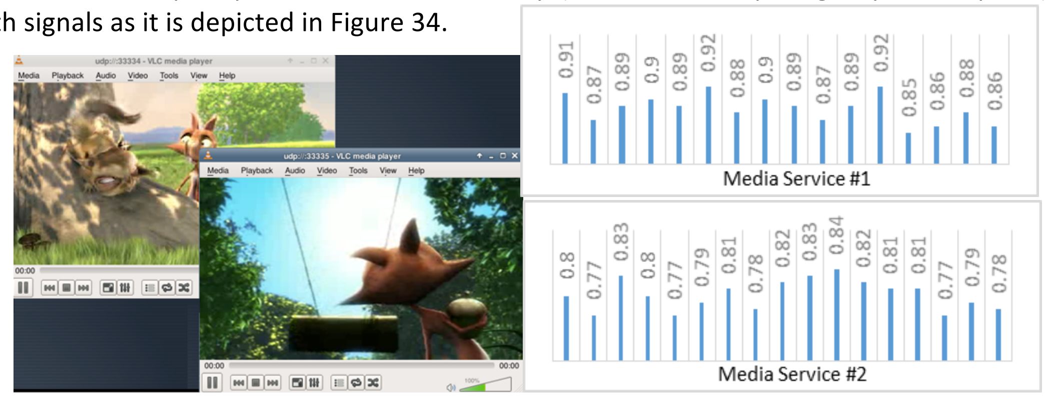


Figure 34. Recovery of the stream quality via transcoding

Finally it should be noted that the responsiveness of the system (measured from the congestion incident until the recovery of the video quality after the traffic steering via the VNFs) was measured approx. at 2 to 3 seconds, depending on the cache configuration of the client at the news aggregator server.

5.1.5.3.2. Conclusions and Added-value

The NFV agility demonstrated in this experiment allows customers to deploy such traffic processing functionalities on-demand in professional satellite terminals, upgrade them and configure/manage them in a unified manner. Resources of virtual appliances can be scaled up and down on-demand, matching the traffic characteristics and customer requirements.

This concept eventually results in a totally new service mix, in which traditional backhauling is coupled with edge processing resources, offered on-demand, as-a-Service. The terminal is essentially transformed to a virtualization-capable remote head-end, able to serve a wide range of use cases.

Last but not least, although the experiment, as described, assumes the use of the satellite terminal by a single customer, virtualization technology allows also multi-tenancy at the edge segment; this means that the professional terminal itself may be partitioned into multiple “virtual terminals”, offered to different customers. This capability can be exploited in scenarios where the satcom operator has already deployed a network of terminals and leases portions of the terminals to different customers. For example, a set of terminals covering a remote village can be leased and shared among two or more mobile operators. This interesting and novel approach demonstrates the power of virtualization technology to introduce new market opportunities and to transform the typical telco value chains.

5.1.5.4. Possible extension of the experiments

The executed experiments have been performed in a single domain PoP and not in a multi-domain environment due to technical problems that rose during the deployment phase of the experiment.

The same experiments could be deployed in multi-PoP and multi-domain environment for demonstrating the advantages of the proposed hybrid coupling between terrestrial and satellite over a multi-domain environment, without however this extension to alter or modify the expected results beyond the ones that have been measured in the single domain experiment.

From a technical perspective further evolutions of SDN technology to facilitate integration with satcom would include:

- Extension of current OpenFlow capabilities to address e.g. specific fields used for the switching process in satcom (such as MPEG-2TS / ATM / ULE / GSE fields in DVB architectures.
- Filtering of events and notifications to allow management of OpenFlow devices from remote controllers via a constrained-bandwidth link, also with high delay.
- Extension of SDN management to embrace radio resource control (especially for multi-GW or HTS systems)
- Better integration with satcom OSS/BSS functions, practices and workflows.

In this framework, further evolutions of NFV technology to facilitate integration with satcom would include:

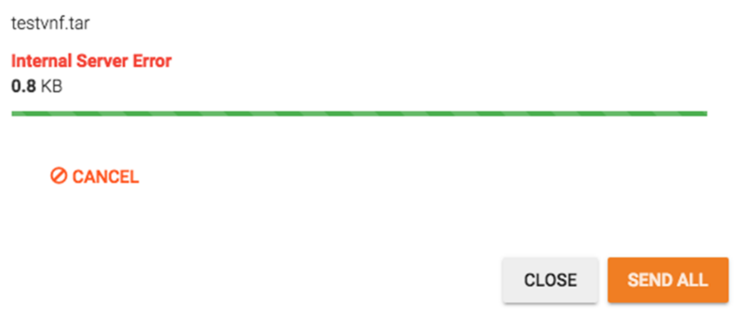
- VNF deployment in compute nodes with very limited resources (e.g. payload or terminal)
- Reconsideration of the Cloud-RAN paradigm to allow efficient distribution to multiple gateways at a very long distance (i.e. to relax bandwidth requirements for backhaul links), for multi-GW configurations. In such scenarios, only specific functionalities should be centralised and not the entire baseband processing chain.
- Better integration with satcom OSS/BSS functions, practices and workflows.

5.1.6. Feedback on the platform and Lessons Learnt

The process of creating the VPN credentials and certificates was straightforward and exactly as described in the wiki documents. Access was acquired immediately and without problems.

5.1.6.1. SoftFire Software Platform

The SoftFire Software Platform enabled the uploading of custom VNF packages and VM images on the platform. Some problems were quickly solved by the platform's tech support and they provided example VNF descriptors and packages Metadata files. It would be nice though that a detailed error description is provided instead of a generic Internal Server Error (Figure 35).


testvnf.tar

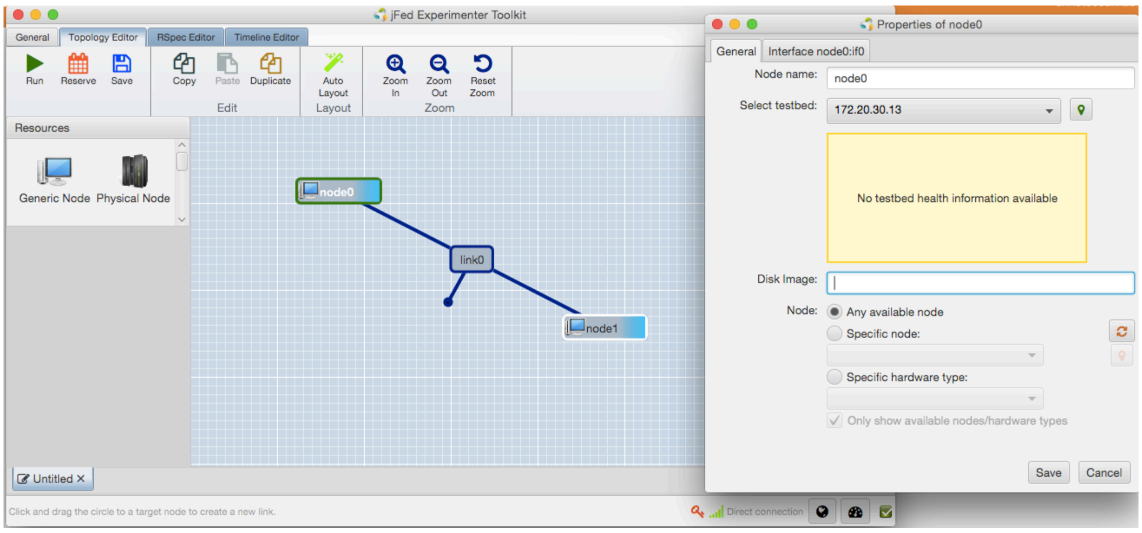
Internal Server Error

0.8 KB

Figure 35 Internal Server Error generic message

5.1.6.2. *jFED*

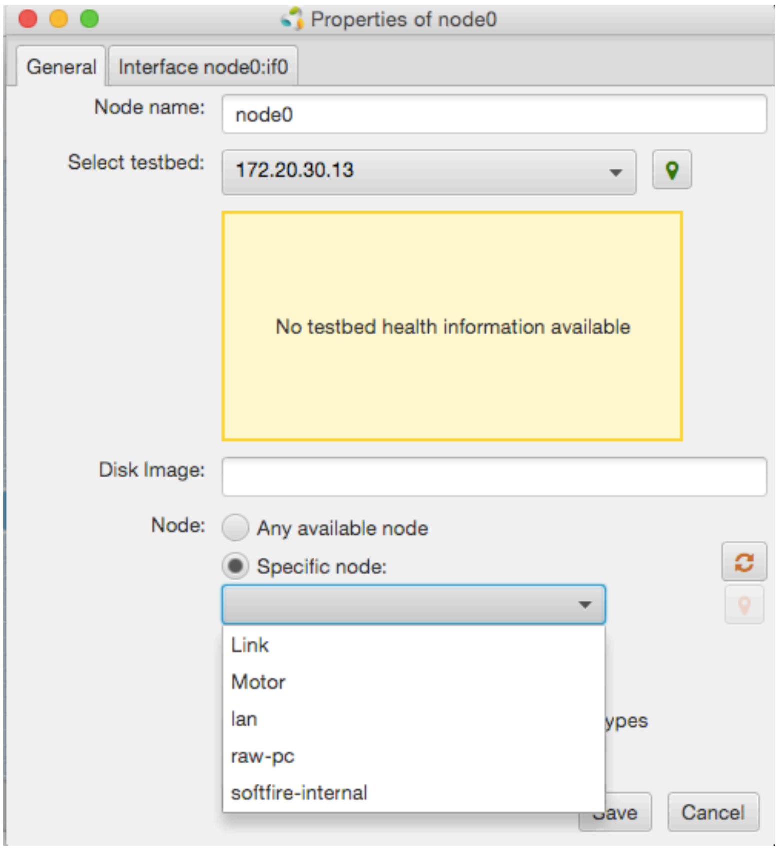
jFED provided an easy to use interface with SoftFire testbeds (Figure 36). Some problems were encountered and they were reported and solved in Redmine issues.



Node name: node0
Select testbed: 172.20.30.13
No testbed health information available
Disk Image:
Node: [✓ Any available node] Specific node: Specific hardware type:
Only show available nodes/hardware types

Figure 36. jFED testbed selection

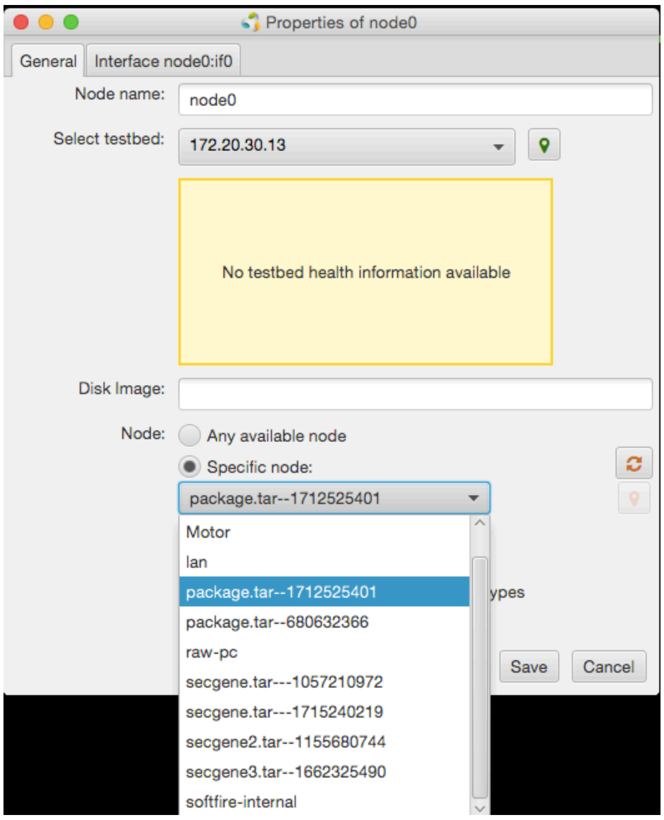
The first one, as reported in Issue 102 (Figure 37) was a confusion about what testbed to use and where the VNF packages were. The tech support quickly provided answers and solutions and we were able to proceed with the preliminary testing of the platform capabilities.



Node name: node0
Select testbed: 172.20.30.13
No testbed health information available
Disk Image:
Node: Any available node [✓ Specific node:]

Figure 37. jFED node selection

The second issue (Figure 38) happened when trying to use a Programmable Switching Device and it generated Java errors, mainly null pointer exceptions. After making issue 103, we were informed that the SoftFire platform currently supports only Generic Nodes so we planned accordingly and used an OpenVSwitch deployment inside a Generic Node to do the switching of packets.



Node name: node0

Select testbed: 172.20.30.13

No testbed health information available

Disk Image:

Node: ( ) Any available node [✓ Specific node:]

package.tar--1712525401

Motor
lan
package.tar--1712525401
package.tar--680632366
raw-pc
secgene.tar---1057210972
secgene.tar---1715240219
secgene2.tar--1155680744
secgene3.tar--1662325490
softfire-internal

Figure 38. jFED VNF selection problem

Another issue was that the VNF packages uploaded in the SoftFire Software Platform could not be found on the list of the specific testbed on jFED. This was documented on issue 105. This and several related to the VNF packages errors were solved in the end.

5.1.6.3. OpenBaton

Deployment through OpenBaton was the next step in the experiment. Some problems were encountered as the platform was not steady in the beginning and were documented by other experimenters in issues 212 and 213. The problems were solved at the end and we managed to successfully run the experiment, take measurements and produce the results. More specifically the problems were:

- ERROR:java.lang.RuntimeException: No EMS for hostname: satellite-gateway-520
- ERROR:org.openbaton.common.vnfm_sdk.exception.VnfmSdkException: Not able to allocate Resources because: org.openbaton.exceptions.VimException: Not found any Image with name: [d7efa53a-4858-49d7-9573-b6d726b77912] on VimInstance vim-instance-uos
- ERROR:org.openbaton.common.vnfm_sdk.exception.VnfmSdkException: Not able to allocate Resources because: org.openbaton.exceptions.VimException: Not launched

VM with hostname satellite-terminal-801 successfully on VimInstance vim-instance-uos. Caused by: org.openbaton.exceptions.VimDriverException: command: POST http://10.5.21.20:8774/v2.1/54abdaadfdc94aae9ddf384e6f6553bf/servers HTTP/1.1 failed with response: HTTP/1.1 400 Bad Request; content: [{"badRequest": {"message": "Can not find requested image", "code": 400}}]soft

5.1.7. Installation Scripts

All machines use Ubuntu 14.04.

Video-server, virtual-transcoder, client:

To install FFmpeg the following commands are needed:

```
sudo add-apt-repository ppa:mc3man/trusty-media
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install ffmpeg
```

OpenSAND:

1. Add the OpenSAND repository
   echo "deb http://packages.net4sat.org/opensand trusty stable" | sudo tee /etc/apt/sources.list.d/opensand.list
2. Update the list of available packages
   sudo apt-get update
3. Install OpenSAND:
   - For Manager (only one host (usually the satellite), used to control the platform)
     sudo apt-get install opensand opensand-manager opensand-collector
   - For all hosts
     sudo apt-get install opensand

ovs-1, ovs-2:

```
sudo apt-get install openvswitch-switch
```

5.1.8. Conclusions

Cloud and Virtualization Networking technologies will have important impacts on future satcom systems. High cost, low resource availability, and conservative architectures that predominate today in the satellite landscape, certainly constitute major obstacles to cross for this family of technologies. On the other hand, many applications could be targeted, and the interests and requirements on cloud and virtualisation networking expressed from all stakeholders do justify additional works in the spatial area.

SDN and NFV, the two main concepts investigated in this work, have different kinds of implication for satcoms. SDN is mainly intended to be implemented at the border of the satcom telecom system, possibly without any impact for the development of its core service in mid-term application, still needing to be integrated with the satcom NMS and OSS/BSS. NFV could have shorter-term applications, related to the operations and management of specific features, wherever they are implemented. For long-term, SDN could also be supported more in depth in satcom.

5.2 Experiment MARS – Level 7 (Mentor : Security Reply)

5.2.1. Executive Summary:

The MARS experiment is focused on building distributed defenses from DDoS attacks, which are well known to be very effective when the attacker is using multiple traffic generator (sometimes referred as zombies) that could attack the target in order to deny a specific service to the real customers. The most challenging part of defense mechanisms is that, while the attack is distributed in the sense that the attacker is using multiple terminals (sometimes more than 10 thousand PCs) in order to carry on the attack the target just sees multiple small flows that are trying to access the service and it's not so easy to understand the real requests/flows from the attacks.

The approach in the MARS architecture is to distribute the defense via probes on the network in order to collect information regarding the status of the flows and enforce some actions in order to drop the bad flows that could cause the denial of services. Therefore the distribution of probes in the network (e.g. the infrastructure of the carrier/ISP) and the enforcers (the elements that would block the packets) are the main goal of the MARS experiment on top of the SoftFIRE infrastructure.

In order to demonstrate the effectiveness of the MARS architecture, the experiment has used two different islands from the SoftFIRE textbed (Fokus and Ericsson) in order to simulate the attack, the target and all the basic elements of the MARS architecture as well as collect data from the experiment.

Moreover, at the beginning of the MARS experiment, some KPI have been selected as goals in order to measure the effectiveness of the MARS experiment as well as to propose new experiments or to refine the technology.

At the end of the MARS experiment, the outcome for the SofFIRE project can be summarized as follows:

- MARS has been validated in a distributed environment
- SoftFIRE has collected information on the use of the testbed, from third party (Level7) and how this can be helpful from a SME in order to speed the validation of novel network architectures
- The MARS experiment needed the access to the network facilities, e.g. routing, and it has introduced some workarounds that can be reused as best practices for new experiments in the new SoftFIRE open calls

5.2.2. Introduction

MARS is a network experiment whose main goal is to test a distributed architecture made of distrusted probes and enforcers as well as a central controller that is responsible to apply some policies that should keep the services in a specified target safe from DDoS attacks.

The MARS experiment has successfully validated the KPI that has been decided at the beginning of the experiment as well as it has produced a video of the experiment.

The main architecture of the MARS experiment can be summarized in the following Figure 39:

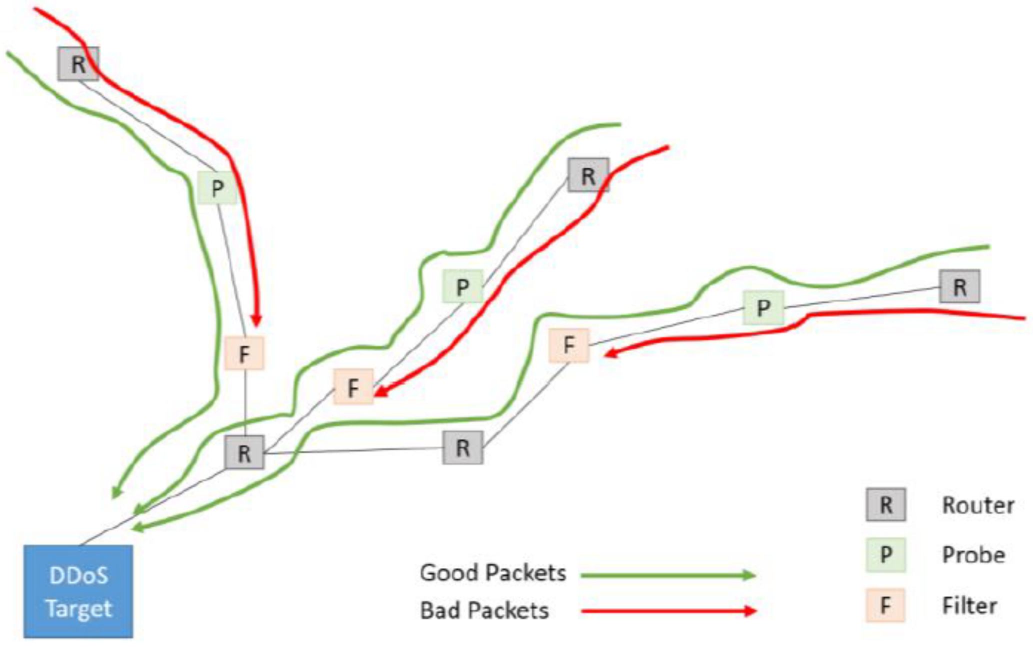


Figure 39 – MARS: Architecture Overview

In Figure 39 above the DDoS target represents a server or a resource on the network that could be attacked from various sources. The sources are distributed and each one of them could send TCP/SYN packets in order to saturate the resources on the target. The Probes are collecting data on the network and sending periodic reports to the Controller that has a general overview of the network status. The Filter (also called Enforcer) has no real intelligence but it will apply the policies that are sent from the Controller to the Enforcers in the network.

The distributed aspect of the architecture has the advantage to block the flows/packets as soon as possible in order to avoid the aggregation in one huge flow that could saturate network resources as bandwidth. It has also the advantage of not consuming resources on the transport network.

In order to measure the effectiveness of the experiment we implemented the MARS architecture and then we measured the KPIs stressing the target with an attack. The results are presented in the following pages.

5.2.3. Project Setup

In order to implement the MARS experiment we used two main islands of the testbed: Fokus and Ericsson. The overall architecture is shown in the following Figure 40:

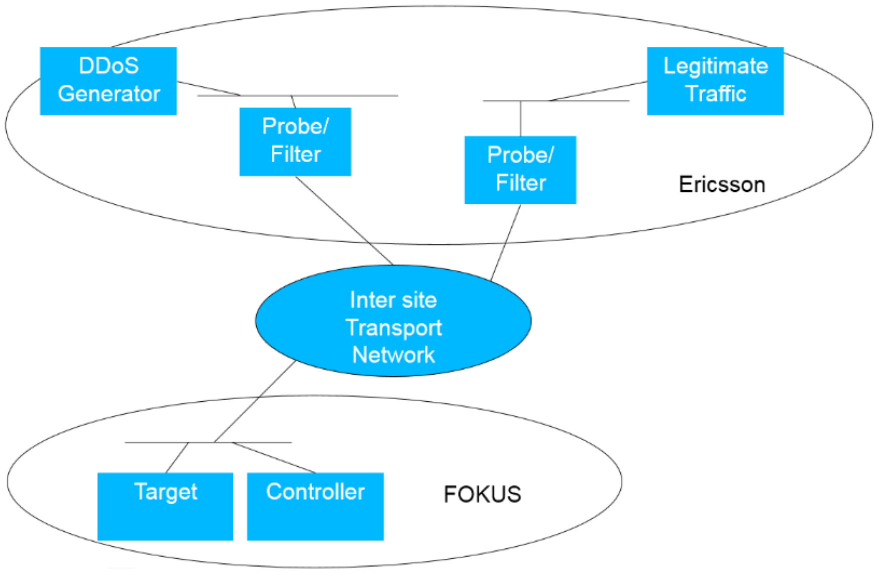


Figure 40 – MARS: Implementation on SoftFIRE Testbed

In the experiment, Fokus has been selected to simulate the Target infrastructure and Ericsson to send bad and good packets to the Target.

Considering that the infrastructure has not the possibility to build a two level network, i.e. create segments that could permit to route the packets via the Probe/Filter machine, the following architecture, based on OpenVPN, has been implemented (Figure 41).

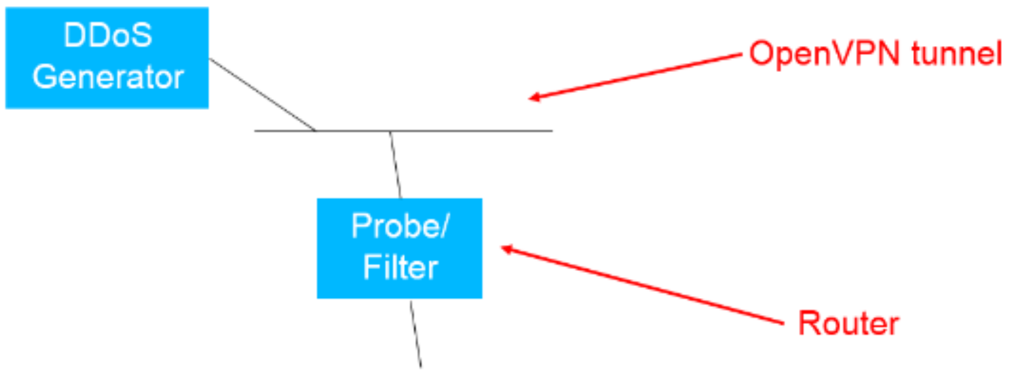


Figure 41 – MARS: Implementation of Routing

5.2.4. Results and Analysis based on KPIs

For the MARS proposal, at the contract negotiation phase, a specific KPI document has been already attached to the MARS contract. This document is also present in the reference section of this report. The results based on the KPI can be summarized in the following Table 5:

	Description	Target Value	Measured Value
KP#1	This KPI measures the percentage of packets that will be recognized by the system, compared to the number of bad packets sent to the target	KPI#1 > 15%	KPI#1 = 100%
KP#2	This KPI measures the percentage of packets that will be blocked by the system, compared to the number of bad packets sent to the target	KPI#2 > 10%	KPI#2 = 99.468%
KP#3	This KPI will measure the percentage of good packets that will be blocked by the system, compared to the number of good packets sent to the target.	KPI#3 < 8%	KPI#3 = 0%
KP#4	This KPI measures the reaction time, i.e. how much time the MARS system takes to react to an attack	KPI#4 < 20%	KPI#4 = 0.489%

Table 5: MARS: KPIs Results

The KPI show that the solution is effective in terms of a defense mechanism in order to discriminate real TCP flows from fake TCP flows (originated by the attacker).

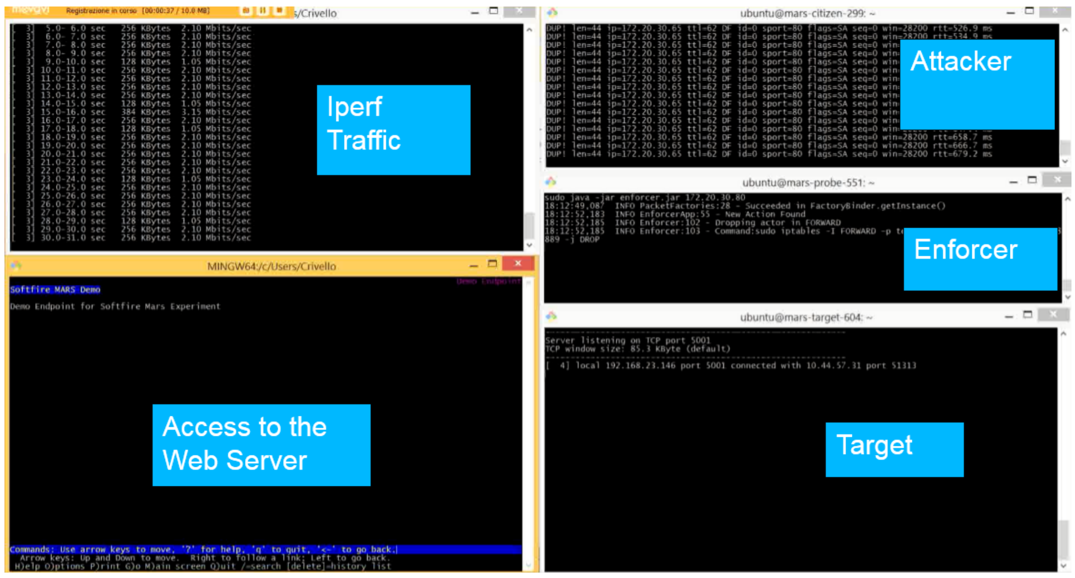
The experiment also shows that a distributed architecture is more effective and needs fewer resources than a centralized architecture.

As main outcome the MARS proposal has produced a demo (which was also a KPI of the proposal) that has been organized in two scenarios:

1. In the Scenario 1 no DDoS solution has been activated.
    a. The service will be affected by DDoS. We start the traffic (Iperf + web page) from the Legitimate Client to the Target. The Target can provide the service to the Legitimate User.
    b. We turn on the DDoS attack.
    c. We show that the web page is no more responsive.
    d. We show that the Iperf from the Legitimate Client is experiencing packet drop.
2. In the Scenario 2 the anti DDoS solution will be activated. The solution will show improvements compared to the Scenario 1. During this Scenario we will also collect some statitics in order to see if we comply with the KPI.
    a. We turn on the anti DDoS solution and we start the traffic (Iperf + web page) from the Legitimate Client to the Target. The Target can provide the service to the Legitimate User.
    b. We turn on the DDoS

c. We show that the web page is still responsive.
d. We show that the Iperf from the Legitimate Client is experiencing few packet drop.

In order to show both the two scenarios, a video for each scenario has been recorded and visually organized as shown in the following Figure 42.


Registrazione in corso [00:00:37 / 10.0 MB]
s/Crivello
Iperf Traffic
ubuntu@mars-citizen-299: ~
Attacker
ubuntu@mars-probe-551: ~
Enforcer
MINGW64:/c/Users/Crivello
Demo Endpoint
Access to the Web Server
ubuntu@mars-target-604: ~
Target

```
[  3]  5.0- 6.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  6.0- 7.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  7.0- 8.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  8.0- 9.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 10.0-11.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 11.0-12.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 12.0-13.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 13.0-14.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 14.0-15.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 15.0-16.0 sec   384 KBytes  3.15 Mbits/sec
[  3] 16.0-17.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 17.0-18.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 18.0-19.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 19.0-20.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 20.0-21.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 21.0-22.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 22.0-23.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 23.0-24.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 24.0-25.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 25.0-26.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 26.0-27.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 27.0-28.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 28.0-29.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 29.0-30.0 sec   256 KBytes  2.10 Mbits/sec
[  3] 30.0-31.0 sec   256 KBytes  2.10 Mbits/sec
```

```
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win=28200 rtt=526.9 ms
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win=28200 rtt=534.9 ms
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win=28200 rtt=658.7 ms
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win=28200 rtt=666.7 ms
DUP! len=44 ip=172.20.30.65 ttl=62 DF id=0 sport=80 flags=SA seq=0 win=28200 rtt=679.2 ms
```

```
sudo java -jar enforcer.jar 172.20.30.80
18:12:49,087  INFO PacketFactories:28 - Succeeded in FactoryBinder.getInstance()
18:12:52,183  INFO EnforcerApp:55 - New Action Found
18:12:52,185  INFO Enforcer:102 - Dropping actor in FORWARD
18:12:52,185  INFO Enforcer:103 - Command:sudo iptables -I FORWARD -p te
889 -j DROP
```

```
Softfire MARS Demo

Demo Endpoint for Softfire Mars Experiment

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.23.146 port 5001 connected with 10.44.57.31 port 51313
```

Figure 42 – MARS: Demo (with MARS turned on)

As reference, a power point presentation for the video has been prepared in order to explain in a more detailed fashion the MARS validation.

The overall architecture will permit to Level7, which is an Internet Service Provider, and its customers to better defend critical services (such as online gaming) as well as general service availability from DDoS attacks.

5.2.5. Lessons learnt

The MARS experiment has shown that a distributed architecture is more efficient than a centralized one and the SoftFIRE project has been very effective in order to validate the basic idea of the proposal.

The main limitation of the SoftFIRE project are related to the not availability (at the time of experiment) of features able to add new network segments in order to build experiments more focused on network validations and simulation.

We think that one of the possible features could be an overlay approach like the one showed by MARS via OpenVPN in order to build “new segments” on top of the distributed testbed.

5.2.6. Summary and Conclusion

The MARS proposal has demonstrated that a distributed approach can be helpful to defend critical infrastructures and generic services from DDoS attacks.

During the experiment, some issues have been resolved thanks to the support from the consortium and via some workarounds.

The target KPIs have been achieved and the MARS architecture has been validated in a real distributed environment.

5.2.7. Technical Annexes according to the Contract

Listed here as:

- KPIs for Managing Attacks from Remote Sources MARS – ANNEX 2 of the MARS contract
- SoftFIRE-Experiment-Validation – PowerPoint presentation v. 2017-02-15-1230
- Demo Video about MARS:
    - MARS_Demo_Scenario_1.avi
    - MARS_Demo_Scenario_2.avi

5.3 Experiment NFV@EDGE – PoliTO (Mentor: TUB)

5.3.1. Introduction

Current solutions for Network Functions Virtualization (NFV) leverage the advantages of IT virtualization to install and operate virtual network functions (NFs) in (remote) data centers. However, this may not be appropriate in some specific deployment scenarios due to (a) possible large latency, (b) limited bandwidth between the end-user and the data center, (c) reliability issues. In these conditions, it may be more appropriate to exploit the existing computing/networking resources at the edge of the network in addition to traditional cloud-based infrastructures, potentially instantiating a service that spans across multiple infrastructure domains, such as a domestic CPE and a remote datacenter.

This experiment, “Network Functions Virtualization at the Edge of the Network” (NFV@EDGE) targets Objective 1 of the First SoftFIRE Open Call and aims at extending the current NFV infrastructure available in SoftFIRE with the capability to control resource-constrained devices, such as home gateways, which are very common at the edge of the network. In this way, an overarching orchestrator can create complex chain of services encompassing network functions running either at the edge or in the cloud, hence bringing in the benefits of edge-based services (e.g., reduced latency, no last-mile bandwidth bottleneck, better reliability) with the ones of cloud-based services (e.g., scalability, efficiency, economy of scale).

NFV@EDGE has achieved its objectives by integrating a new NFV-capable platform, the Universal Node (UN), under the OpenBaton (OB) toolkit. The UN is a compact software orchestrator targeted to resource-limited devices such as domestic/SOHO residential gateways. Its integration in SoftFIRE enablee OpenBaton to create complex NFV services by instantiating the required NFs partly on edge-located nodes and partly on datacenters, in a transparent way with respect to the service requester.

Finally, NFV@EDGE has validated the proposed approach by running a set of experiments on the SoftFIRE infrastructure, simulating remote tenants asking for NFV services in different topological conditions and evaluating the achieved performance with and without edge-based nodes, in order to assess the benefits of edge-based services on a real geographically distributed infrastructure.

5.3.2. Project Setup

NFV@EDGE proposes the integration of the UN in the SoftFIRE testbed and the validation of the "NFV at the edge of the network" approach through an appropriate set of experiments.

The following sections will detail how the project has been organized in different activities, and the KPI that have been defined to measure the progress and the expected outcome.

5.3.2.1. Activity breakdown

From the technical standpoint, this project has been organized in the following four phases.

5.3.2.1.1. Integration of the UN in the SoftFIRE architecture

In SoftFIRE, NFV services are managed through the OpenBaton orchestrator, which is a modular software that can control multiple and heterogeneous infrastructure domains through the sp called Virtual Infrastructure Managers (VIM), according to the ETSI terminology. NFV@EDGE requires the extension of the OpenBaton platform with a new VIM plug-in (as shown in Figure 43) that is able to control the Universal Node. In a nutshell, this software module implements the basic actions that are needed by OpenBaton to control the underlying infrastructure (e.g., create/delete network, start/stop a virtual machine), translating high-level commands into the ones supported by the involved domain. In order to guarantee the seamless integration of the UN architecture in the existing software framework, the new UN VIM plugin makes use of many OpenStack concepts, such as user/group permissions, availability zones, etc., adapting them to the UN domain. This will guarantee that, from the experimenter standpoint, the UN appears just like another OpenStack domain, hence hiding its different internal architecture.

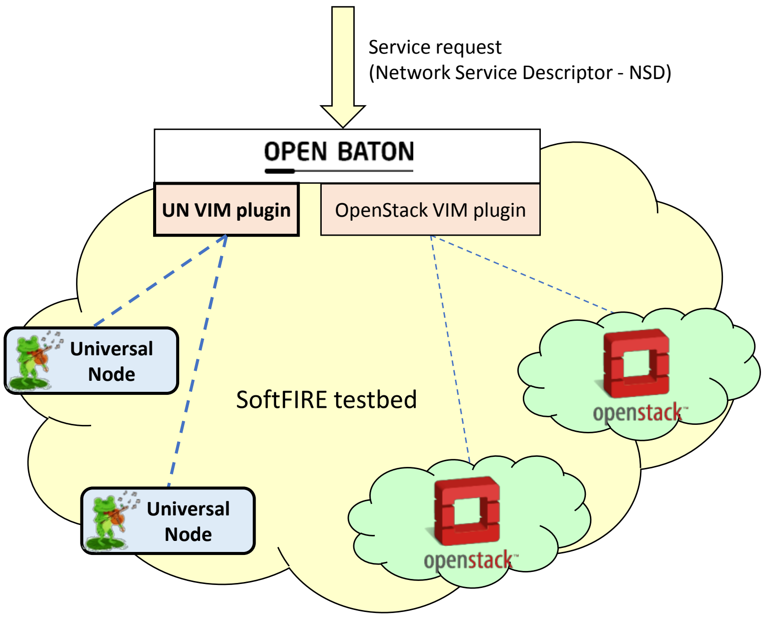


Figure 43. Overall architecture of NFV@EDGE and its relationship with existing modules. Modules developed in the project are shown in thick black.

5.3.2.1.2. Integration of the UN in the SoftFIRE testbed

This activity has been dedicated to the integration of the UN in the SoftFIRE testbed. This requires (*i*) to update the master installation of the OpenBaton orchestrator with the new VIM plugin developed in the previous activity, and (*ii*) integrate a couple of UN in the SoftFIRE testbed in order to run the experiments.

5.3.2.1.3. Creation of the required VNFs running on the SoftFIRE infrastructure

VNFs are not simple virtual machines with some network functions running in it, but they need to be extended with the proper software in order to be controlled by the OB orchestrator/VNFM. This activity takes care of creating the proper VMs that are needed to deploy the service that will be used in the validation phase.

5.3.2.1.4. Validation of the "NFV at the edge" approach

This activity has to set up different NFV services consisting of many network functions, validating at least the two scenarios depicted in Figure 44. The first scenario consists in offloading the VPN client to the network (e.g., his residential gateway), hence allowing users to connect to their corporate VPN server without having to install anything on their user-devices, hence enabling VPN access from any device, e.g., also a smart TV. This scenario is very close to the KPI-1 mentioned in this experiment and it guarantees that the integration of the UN in the SoftFIRE testbed has been successful.

The second scenario validates the difference in terms of performance when the VNFs required to set up a domestic Internet access (i.e., a LAN switch, DHCP, NAT, firewall) are installed in different portions of the infrastructure, in order to quantify the advantages of the NFV@EDGE scenario. The second scenario is currently impossible in the POLITO premises, as it requires the availability of a geographical network. This represent one of the benefits of SoftFIRE in this experiment, which enables to carry out real measurement in a truly distributed geographical network.

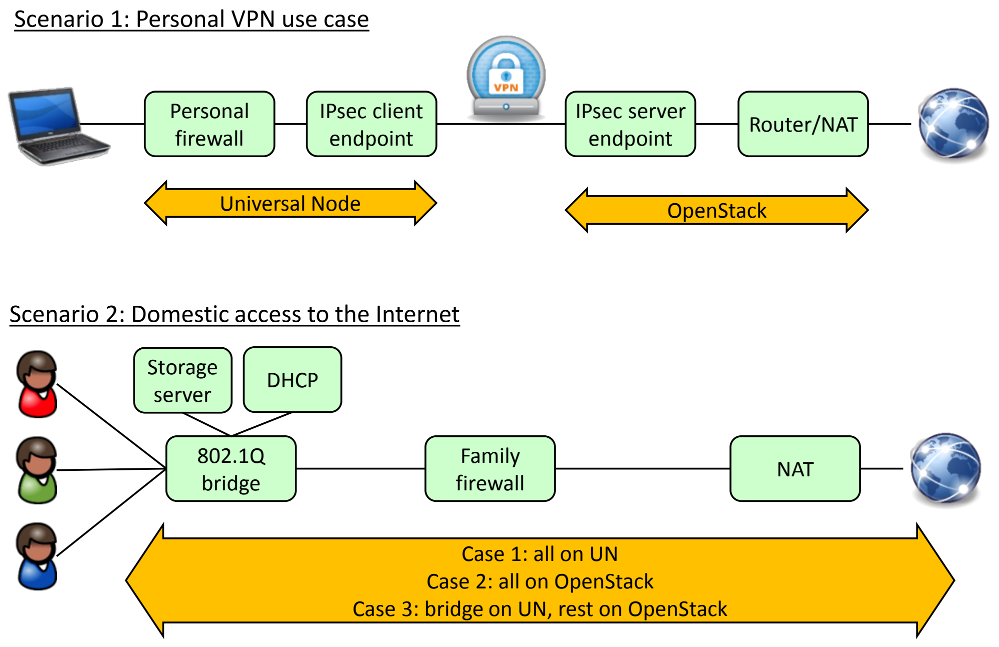


Figure 44. Scenarios used in the validation.

5.3.2.2. KPIs

The NVF@EDGE experiment defines the following KPIs (Table 6 and Table 7).

Table 6. NFV@EDGE KPI-1.

KPI-1: UN integrated with SoftFIRE and running at POLITO	
Description	Type
Two Universal Nodes installed at POLITO premises are up and running and integrated with the SoftFIRE infrastructure.	Boolean (passed / not passed)
Means of verification	
This KPI is verified by instantiating two cascading VNFs (one in the UN domain, the other in the OpenStack domain) through the FITEagle interface. The two VNFs will have only a software bridge installed on all the virtual ports and all the traffic that enters from one side of the chain is delivered to the other end of the chain. This simple setup guarantees that the integration UN-SoftFIRE has been completed; hence, NFV@EDGE is ready to move to the second phase (experimentation).	

Table 7. NFV@EDGE KPI-2.

KPI-2: NFV@EDGE guarantees better performance than pure cloud service delivery	
Description	Type
The setup of a service installed partly on the edge device (UN) and partly in the datacenter shows that the NFV@EDGE approach can guarantee better performance than a fully datacenter-based service delivery.	Quantitative: measurement of the performance (throughput, latency)
Means of verification	
This KPI is verified by instantiating a service in which a first portion is executed by the edge device (UN) while a second portion runs in the datacenter and measuring the latency and throughput between the clients and the deployed services. The measurement is repeated also in different conditions such as when all services are running the edge when and services are running in the cloud.	

5.3.3. Results and Analysis based on KPIs

This section describes the work done to prepare the experiment and the obtained results.

5.3.3.1. Integration of the UN in the SoftFIRE architecture

This section presents the work related to the integration of the UN in the SoftFIRE software environment.

5.3.3.1.1. Universal Node

The Universal Node, depicted in Figure 45, is a compact service orchestration platform, mostly written in C, which offers the possibility to compose network functions (NFs) in arbitrary service graphs and, in the end, deliver virtualized services. The service graph described through a high-level formalism, the Network Function Forwarding Graph (NFFG), which illustrates which VNFs and what endpoints are involved and how they are connected between them. An endpoint represents an entry/exit point in/from the NFFG, allowing a VNF to be connected with a generic element; in particular, the endpoints defined in the UN architecture allow to connect a VNF with a physical interface of the host, with an element of a different NFFG or with the stack of the host machine.

The UN can be considered as a "clean slate" architecture, in which one of the key concept is the strong separation between infrastructure-level commands (NFs, links, endpoints) and service-level configurations. In a nutshell, the UN is only responsible of deploying NF (and, in general, managing its lifecycle), and creating the proper traffic steering connections between the above modules. All the rest, such as the proper IP addresses that are required for the service to operate, are completely out of scope and must be handled with other modules. Only MAC addresses are considered as belonging to the infrastructure; hence they can be assigned to virtual network interfaces (vNICs) through the NFFG.

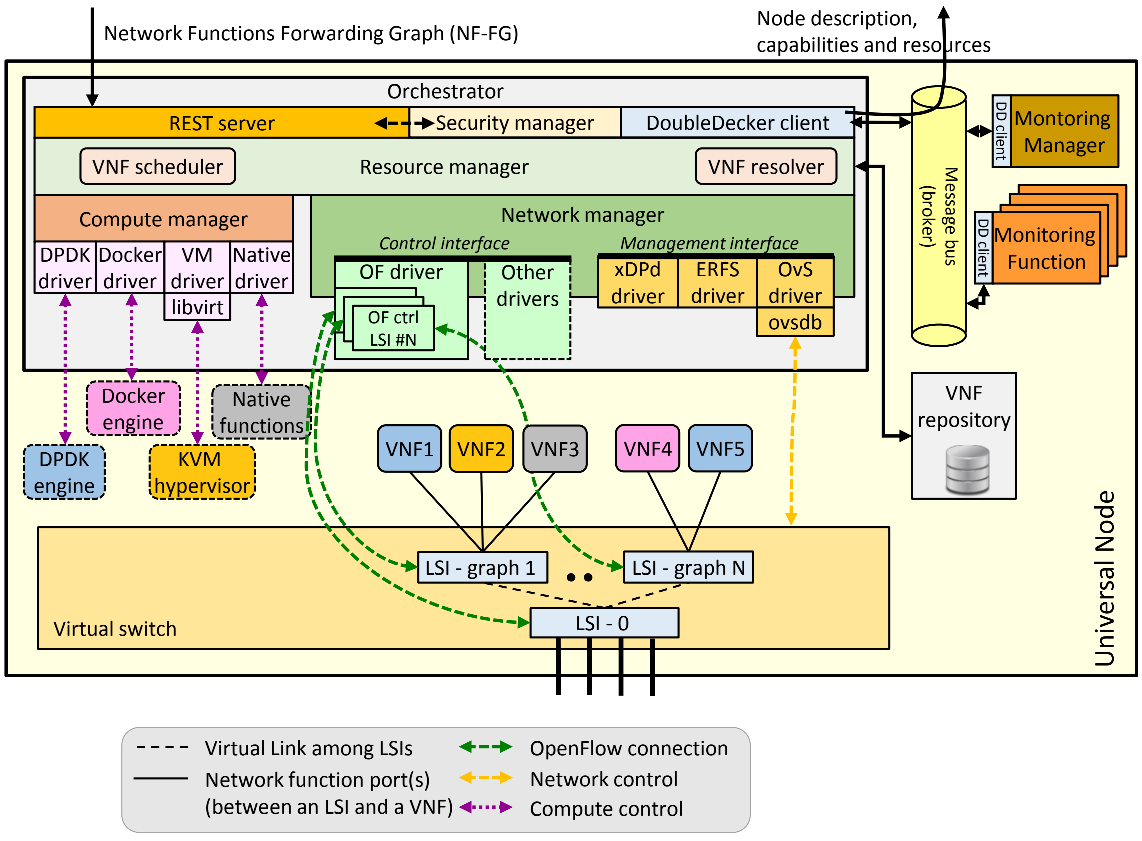


Figure 45. Universal Node: architectural overview.

5.3.3.1.2. The configuration and management model in the UN

In order to have a fully running service, the UN must be associated with a set of components that are fully dedicated to configuration purposes.

The overall architecture, as depicted in Figure 45, is composed of four main modules. First, the UN analyzes the NFFG and implements the requested service, hence instantiating the required VNFs and setting the proper traffic steering primitives to connect them together (and to the endpoints). Second, the configuration service is in charge of both the runtime configuration and to export the run-time state of VNFs. In a nutshell, it provides a REST API that can be used to either set or read a configuration to any controlled object (e.g., VNF, but also the UN itself). Third, a flexible datastore keeps different kind of data, such as the VNF images and the associated templates, the YANG data models of any supported object, users and groups permissions, and more. Finally, a message bus, based on DoubleDecker[1] that connects the configuration service with all configuration agents that are running on the different objects.

The choice of a message bus is one of the key features of the system, as it supports the *publish* and *subscribe* primitives, hence decoupling producers from consumers of the information. In fact, this enables the generation of information that is sent automatically to the entities that need it (without knowing who they are), and to receive information no matter who will generate it.

The overall configuration architecture exploits YANG data models, associated to any managed object (e.g., VNFs), that describes the object in terms of configuration and state. For instance, in case of a firewall VNF, the configuration includes the policy rules that define which traffic is allowed and which one is blocked, while the state can be represented by the number of sessions that have been blocked by each rule. Data models enable the definition of a generic mechanism that can be used to interact with the network functions, by *setting* or *getting* data associated to the YANG model, which in turns become write/read commands toward the managed object. Given the generality of the YANG model, the above architecture is used to control a wide range of objects, such as UN basic parameters (e.g., IP addresses associated to any interface), VNF configuration (e.g., set the range of IP addresses to be used by a VNF) and status (e.g., the IP addresses already assigned to requesting clients).

Each controlled object includes an agent, e.g. installed in each VNF, in order to interact with the other configuration components through the message bus and translate the configuration primitives, transmitted on the bus, into the actual configuration commands (e.g., local configuration file). In fact, the agent sends/receive messages through the DoubleDecker bus, either publishing information, or subscribing information that can be possibly (and asynchronously) generated by other components. In addition, DoubleDecker supports also direct messages that are delivered directly to the specified recipient, e.g., useful in case we need to know exactly a piece of information stored in a given VNF.

The delivery of management and control information requires the availability of a (virtual) network infrastructure, but the architecture of that network is not specified at all. The suggested architecture, however, includes a dedicated network for control and management purposes, where only the ctrl/mgmt components are attached. For instance, this requires a dedicated vNIC on all the VNF, which is used by the above-mentioned agent. Therefore, many services will require the setup of multiple networks (or chains), one dedicated to the data plane (handled by the softswitch in Figure 46), another dedicated to the ctrl/mgmt plane,

[1] http://acreo.github.io/DoubleDecker/.

devoted to this task (shown by the yellow components in Figure 46). In some sense, it appears such as the vNIC dedicated to the ctrl/mgmt task is somehow hidden to the service requester (a.k.a. tenant).

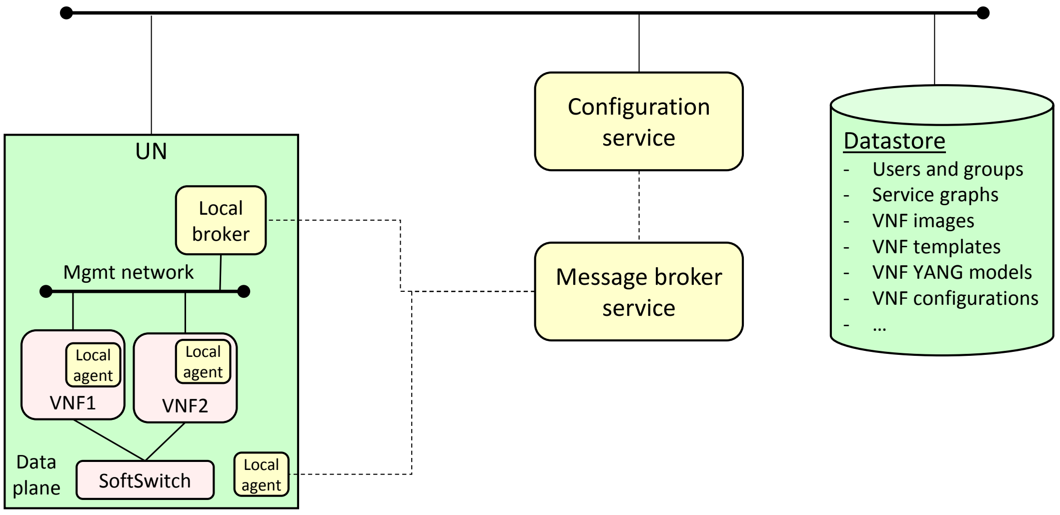


Figure 46. The UN and the overall configuration architecture.

5.3.3.1.3. Overall view of the integration of the UN in the OB architecture

Given the structure of the UN and the companion necessity of the additional services for configuration purposes, the integration of the UN in the OB requires the setup of multiple components, as shown in Figure 47. In this picture, the new components have been depicted in green, while existing ones (i.e., the one already belonging to the SoftFIRE testbed) are depicted in yellow. An example of a possible service graph is depicted as well, with the corresponding mapping in the infrastructure. The service graph, in fact, originates a very different implementation at the infrastructure layer, with three additional VNFs (depicted in light blue) required to support the service itself (e.g., LAN emulation, router/NAT toward the Internet), other three VNFs dedicated to the management and configuration tasks (depicted in pink), while apparently the "requested" service includes only two VNFs (orange components).

In this particular mapping example, the UN is able to accept services through a single NIC (mimicking the typical case of residential gateways, which feature a single NIC toward the operator network); however, at the time of writing, three public IP addresses are required, as depicted by the black spots in the Figure. For instance, the first is used to connect the requested service to Internet, the second enables the reachability of the services running in the management network (required to configure the VNFs), while the third is used for the general management of the UN itself, e.g., to send the service graph (NFFG) and to interact with the UN orchestrator.

More details about the mapping algorithm will follow in the next Sections.

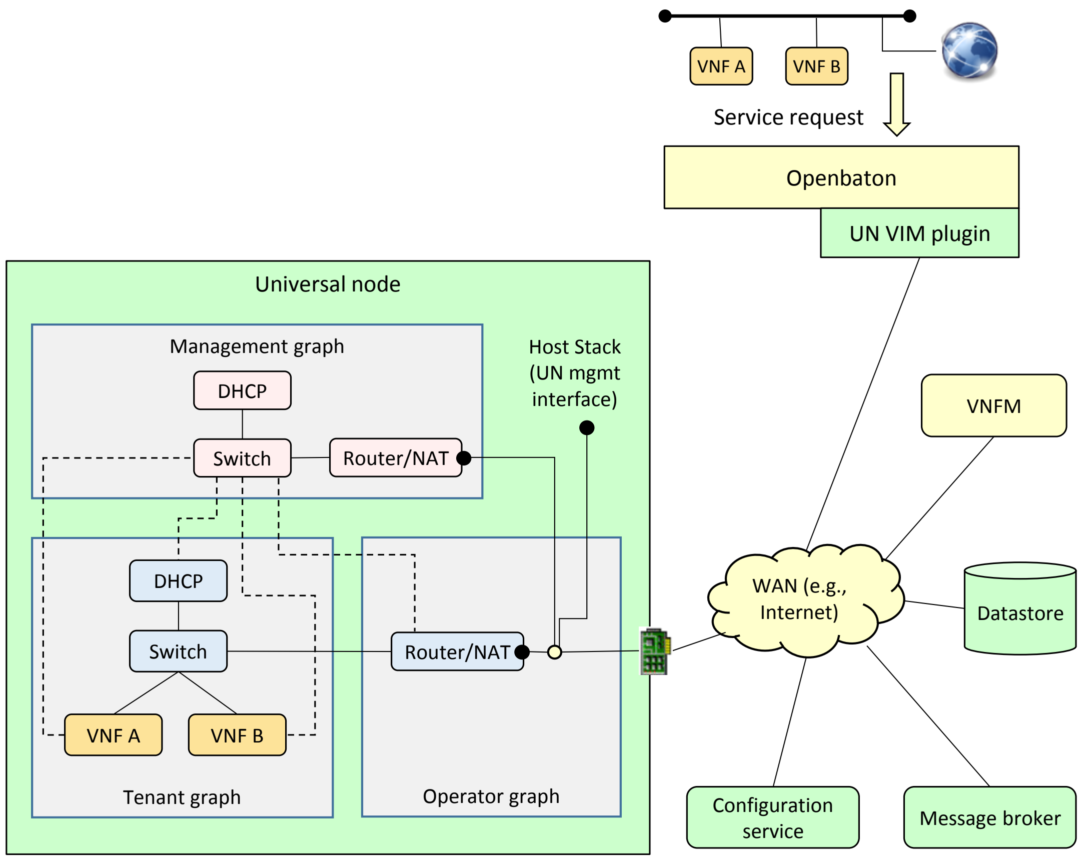


Figure 47. Integration of the UN in the OB architecture.

5.3.3.1.4. Management operations in ETSI MANO

The interaction of the VNFs with the VNFM, as defined by the ETSI MANO model, can be considered a sort of operation that belongs to the management space, even though the term "management" is not actually used in the ETSI specifications.

Given the similar nature of the NF/VNFM interaction and the necessity to configure the VNFs with a separate network (in the UN), we decided to integrate the above management operations with the management network that is already present in the service deployed on the UN.

However, this mapping is not completely transparent to the external world. In fact, although the service requester (a.k.a., "tenant") does not have to make any change to its service (e.g., the NSD does not need to be changed in order to define the management network), some changes are required for the VNF creator, who has to explicitly introduce an additional vNIC dedicated to management in the VNF image. However, this vNIC will be automatically configured by the UN VIM plugin, which currently assumes that this interface is the first one (e.g., eth0) attached to the VM itself. In addition, the UN VIM plugin has to modify the generic data, coming from the VNFM, that is passed to any VNF at boot time and that is read by the *cloud.init* daemon. This data will automatically be modified by the UN VIM plugin, adding the commands needed to configure an ad hoc routing rules that forces the VNF to forward all the traffic toward the VNFM through the management interface.

In the end, the management network provides a connection to both the VNFM and the WAN, enabling the VNF to download all the software required for the service execution[2], even if the service designed by the tenant has no Internet connection.

An alternative approach that was implemented in the initial phase of the project consisted in handling the connection to the VNFM through the same vNICs used for the data plane. In that case, the *cloud.init* scripts configured the Internet connectivity (hence, a default route) through the data interface, and, in case the service graph specifies that the VNF does not have to be connected to the Internet, that configuration has to be cleaned up when the VNFs was fully configured and running. However this approach forced to use the same path for both user data and management task, hence it was replaced with the one presented above that looks cleaner.

Finally, it is worth mentioning that the initial configuration script, returned by the VNFM, is not domain-specific, hence the same VNFM can be used to handle different domains. In fact, the modifications to that script that allow the service to be instantiated on the UN are done transparently by the UN VIM plugin, hence without requiring any change to the original script in the VNFM.

5.3.3.1.5. UN VIM plugin development

The VIM plugin is the component in charge of translating the OB primitives into infrastructure-specific commands. Since OpenStack is the most used infrastructure-level controller, the set of the VIM plugin methjods that have to be implemented are derived from the concepts that are available in OpenStack. As a consequence, most of the VIM plugin primitives are just a call to the corresponding API in OpenStack.

However, the Universal Node has a strong separation between the infrastructure-level primitives (e.g., NF, links, endpoints…) and the configuration (e.g., IP addresses); as a consequence, the mapping can be more complex than the case of OpenStack. In some cases, this requires to interact with multiple components (e.g., UN orchestrator, configuration server, etc.) in order to implement the requested service.

The following Table 8 lists all the methods supported by the UN VIM plugin, with a brief explanation of its implementation. In case of a complex implementation, a dedicated subsection is provided.

Table 8. List of OB primitives supported in the UN VIM plugin.

	Name	Description
1	launchInstance	It creates and asks for the launching of an instance of the VNF and immediately returns. Refer to subsection 5.3.3.1.5.3 for more details.
2	listImages	It lists all the available images that this VIM instance can use.

[2] In OB, the preferred way of operation is to start all the VNFs with the same base image, then a set of script is used to download all the software required for the VNF to perform its activities, e.g., through a set of *apt-get* commands.

3	listServer	It lists all the VNFs that have already been deployed. Refer to subsection 5.3.3.1.5.3 for more details.
4	listNetworks	It lists all the networks that are already active in the target domain.
5	listFlavours	It lists all the flavors supported by the current VIM instance. This method is not supported by the UN, hence it is emulated by the VIM plugin.
6	launchInstanceAndWait	It creates and asks for the launching of an instance of the VNF and waits until it is launched. Refer to subsection 5.3.3.1.5.3 for more details.
9	deleteServerById	It asks for the deletion of the VNF with a given ID and immediately returns.
10	deleteServerByIdAndWait	It asks for the deletion of the VNF with a given ID and waits until it is deleted.
11	createNetwork	It creates a new network. Refer to subsection 5.3.3.1.5.1 for more details.
12	createSubnet	It creates a new subnet. Refer to subsection 5.3.3.1.5.2 for more details.
13	updateSubnet	It updates some subnet parameters.
14	getSubnetsExtIds	It returns, for each subnet, its ext id.
15	deleteSubnet	It deletes the subnet with a given ID.
16	getNetworkById	It returns the network with a given ID.
17	getQuota	It returns the quota assigned to the tenant (not supported by the UN, emulated by the plugin).
18	getType	It returns the type of the VIM.

5.3.3.1.5.1 Network creation

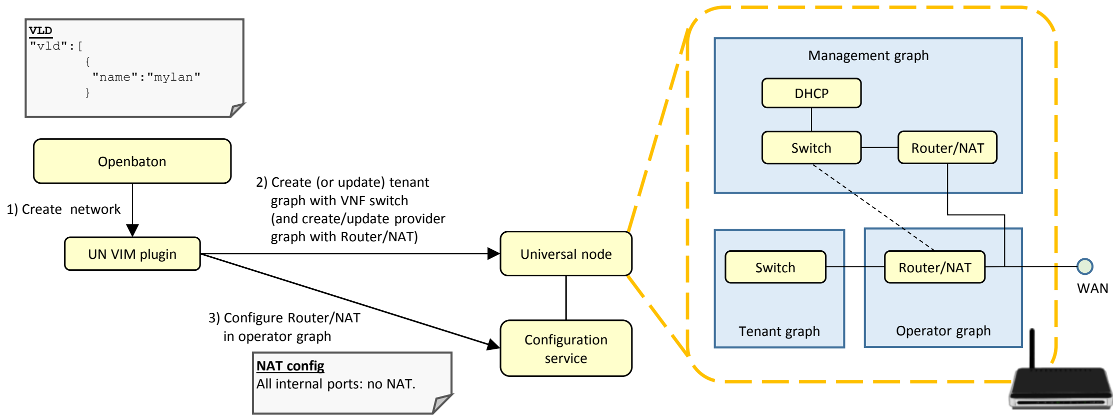
The UN does not have the concept of "network"; it just deploys VNFs and connects them following the rules specified into the NFFG.

As shown in Figure 48, a network is emulated by the UN by setting up with a VNF containing a bridge (e.g., Linuxbridge or OpenvSwitch working in "normal" mode), which is connected to all the VNFs attached to the network itself. This require that the UN VNF repository contains a special "bridge VNF", whose name is used by the UN VIM plugin to transform the incoming service graph into the proper NFFG to be deployed, which is a simple graph containing the above bridge VNF.

Unfortunately, the process of creating a network is not so simple, because there may be the case of VNFs attached to that network have to connect to the Internet. However, this situation, which happens when a VNF is associated to a “floating IP”, is currently unknown, because first we need to create the network, then attach the VNFs, hence we do not know yet whether that network has to be connected to the Internet or not.

For this reason, when a network is created, the “bridge VNF” is always connected to a Router/NAT VNF[3] that can potentially provide Internet (or, in general, WAN) connectivity as shown in **Fehler! Verweisquelle konnte nicht gefunden werden.**. However, given that it is still unknown whether the Internet connectivity is needed, the Router/NAT VNF is automatically configured to blocks all the incoming traffic toward the tenant's LAN switch. This configuration is modified by the UN VIM plugin when the request for a new VNF with a floating IP is received.

It is worth noting that OB has no visibility on the bridge VNF, as the UN VIM plugin exposes to OB exactly the network service that has been requested, without detailing the actual implementation.



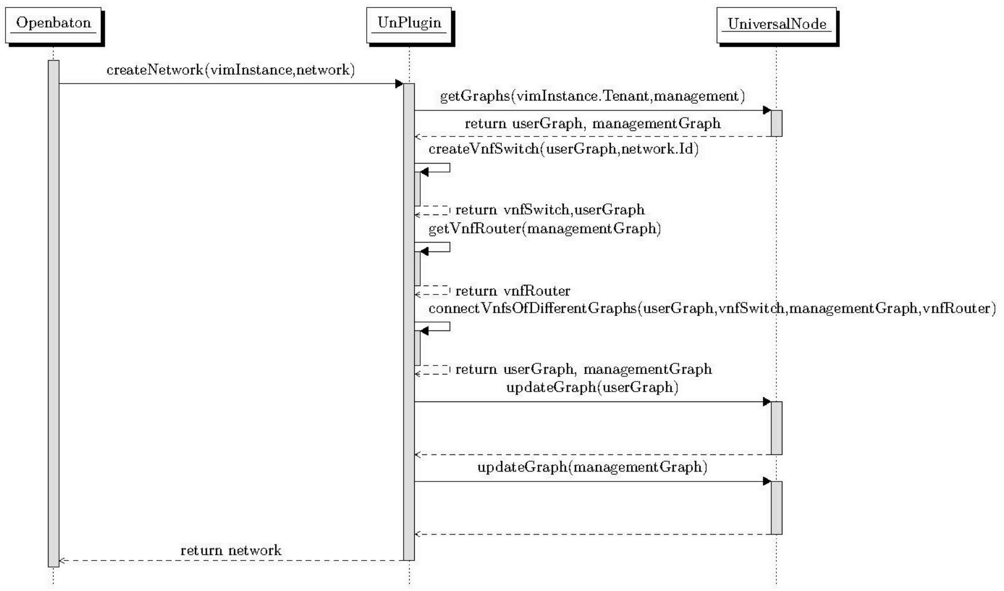
VLD
"vld":[{ "name":"mylan" }
Openbaton
1) Create network
UN VIM plugin
2) Create (or update) tenant graph with VNF switch (and create/update provider graph with Router/NAT)
Universal node
3) Configure Router/NAT in operator graph
NAT config
All internal ports: no NAT.
Configuration service
Management graph
DHCP
Switch
Router/NAT
Switch
Router/NAT
Tenant graph
Operator graph
WAN

Figure 48. Network creation[4]: example.

As shown in Figure 49, the actual implementation is definitely more complex; the UN VIM plugin performs a not negligible number of operations when receiving a “create network” request from OB before returning an answer. First of all it queries the UN in order to retrieve both the tenant and the management graphs, then the plugin modifies them locally adding a new switch in the tenant graph and creating a link between it and the router of the management graph. After communicating such updates to the UN, the plugin can finally return OB an object representing the network just instantiated.

[3] Since the Router/NAT VNF that provides Internet connectivity has to serve the graphs of all the tenants, it is only created once at the beginning, when the first request to create a network is received.

[4] The example depicted in this Figure originates by the simple VLD shown in the picture. In this respect, OB implements only one type of the link defined in ETSI MANO, i.e., the *E-LAN* link, which is a local area network associated with an IP subnet. Therefore, that simple VLD will originate both the CreateNetwork() and CreateSubnet() calls in the VIM plugin; as a consequence, the same input VLD is found in both Figure 48 and Figure 50, which represent two consecutive steps of the same operation.



Openbaton
UnPlugin
UniversalNode
createNetwork(vimInstance,network)
getGraphs(vimInstance.Tenant,management)
return userGraph, managementGraph
createVnfSwitch(userGraph,network.Id)
return vnfSwitch,userGraph
getVnfRouter(managementGraph)
return vnfRouter
connectVnfsOfDifferentGraphs(userGraph,vnfSwitch,managementGraph,vnfRouter)
return userGraph, managementGraph
updateGraph(userGraph)
updateGraph(managementGraph)
return network

Figure 49. Network creation: actual workflow.

5.3.3.1.5.2 Subnet creation

When OB sends a “create subnet” request to a VIM plugin, it expects the VIM to choose an IP address for the default gateway and that, from that moment on, the VIM will assign an appropriate IP address to all the VNFs that will be connected to such a subnet.

As shown in Figure 50, the UN VIM plugin translates such a request in a sequence of 3 operations; first of all it creates a DHCP and attaches it to the switch associated to the subnet, hence providing the full emulation of an IP LAN. Second, it asks the configuration service to push the IP configuration parameters in the DHCP VNF. The above parameters (IP subnet range, netmask, IP address of the default gateway) are decided by OB and are required to configure future VNF (e.g., hosts/servers) that will be attached to that LAN. Finally, it configures the appropriate IP address to the internal vNIC of the router/NAT VNF in the provider graph, which will be needed to provide Internet connectivity to that LAN.

Currently OB does not specify, per each VNF, which subnet has to be attached to; hence, in case multiple subnets are associated to the same network, the VNF will randomly obtain an IP address belonging to a subnet rather than another.

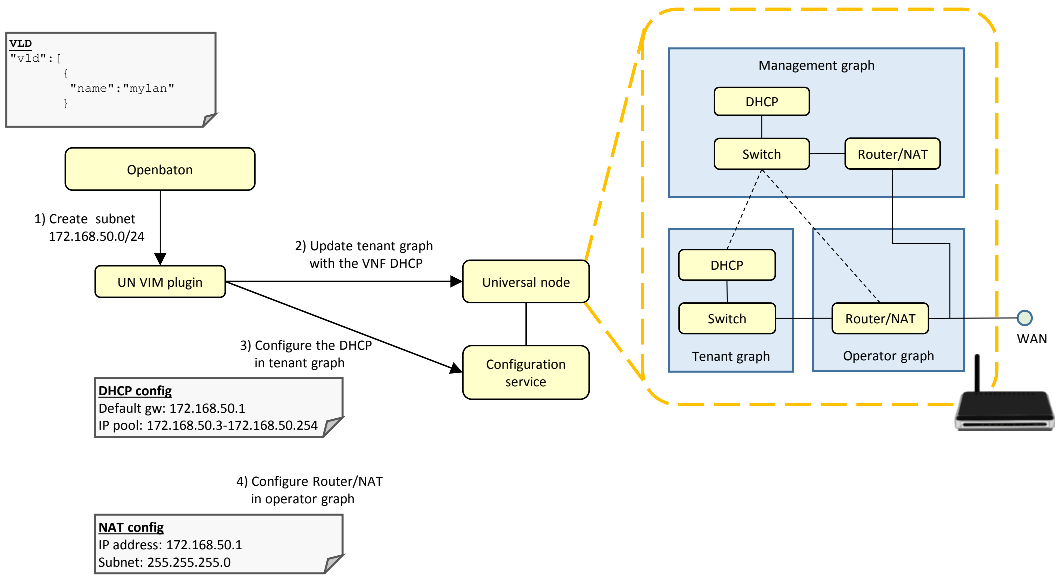


Figure 50. Subnet creation: example.

The actual workflow is depicted in Figure 51. First, the UN VIM plugin queries the UN to retrieve both the tenant and the management graphs. Second, it modifies them locally by adding a new DHCP in the tenant graph connected to the LAN switch and by creating a connection between the DHCP and the management graph, which enables the configuration of that VNF. After sending the updated graphs to the UN, the plugin configures both the LAN DHCP (with the parameters chosen by OB) and the internal interface of the Router/ NAT[5] (with an IP address belonging to the configured subnet, which will be also the default gateway for the entire network). Finally, it returns to OB an object representing the subnet just instantiated.

[5] In the mapping model between OB and the UN, we create a single Router/NAT in the operator graph, which can be connected to multiple tenant graphs by means of a dedicated vNIC. Hence, each tenant graph sees its own default gateway, configured with the IP address chosen for that subnet.

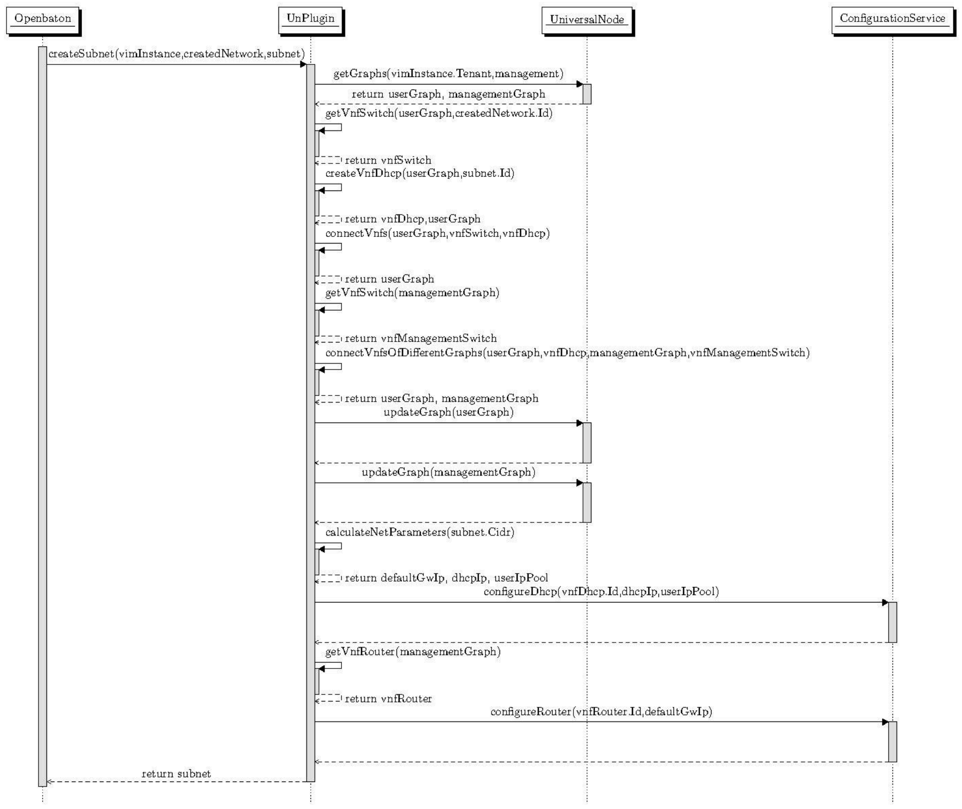

Openbaton
UnPlugin
UniversalNode
ConfigurationService
createSubnet(vimInstance,createdNetwork,subnet)
getGraphs(vimInstance.Tenant,management)
return userGraph, managementGraph
getVnfSwitch(userGraph,createdNetwork.Id)
return vnfSwitch
createVnfDhcp(userGraph,subnet.Id)
return vnfDhcp,userGraph
connectVnfs(userGraph,vnfSwitch,vnfDhcp)
return userGraph
getVnfSwitch(managementGraph)
return vnfManagementSwitch
connectVnfsOfDifferentGraphs(userGraph,vnfDhcp,managementGraph,vnfManagementSwitch)
return userGraph, managementGraph
updateGraph(userGraph)
updateGraph(managementGraph)
calculateNetParameters(subnet.Cidr)
return defaultGwIp, dhcpIp, userIpPool
configureDhcp(vnfDhcp.Id,dhcpIp,userIpPool)
getVnfRouter(managementGraph)
return vnfRouter
configureRouter(vnfRouter.Id,defaultGwIp)
return subnet

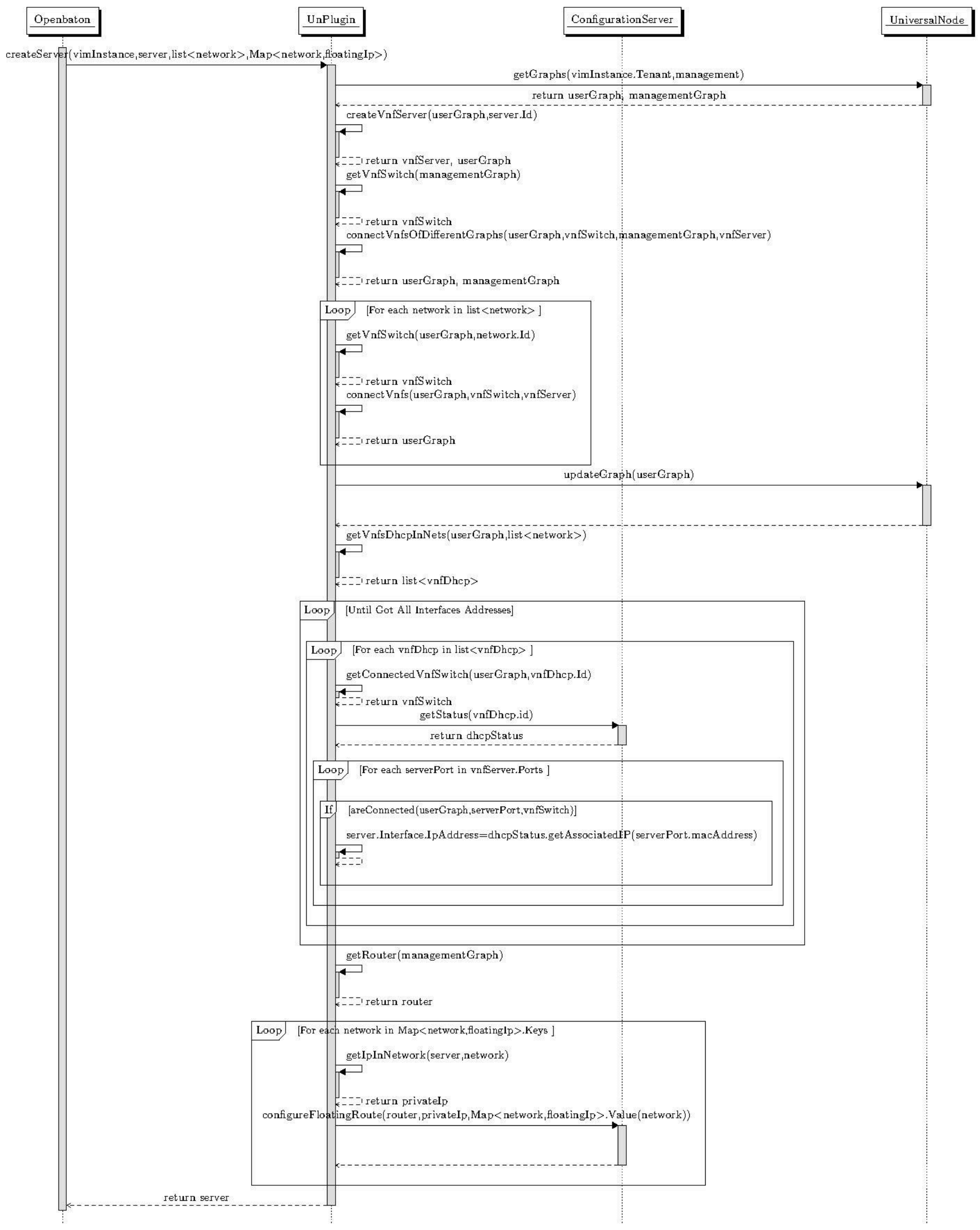
Figure 51. Subnet creation: actual workflow.

5.3.3.1.5.3 VNF (a.k.a. Server) creation

When OB sends to a "create server" request to the VIM plugin, it expect the infrastructure controller to instantiate the required VNF and create the proper connection between the VNF ports (i.e., vNICs) and the networks they are attached to. Moreover, the VNF is also connected to the management network in order to provide (*i*) a way to configure it, if needed, and (*ii*) a path toward the VNFM. Finally, in case some vNICs are associated to a "floating IP", it updates the configuration of the Router/NAT in the operator graph in order to allow the internal IP address to be reachable from the Internet.

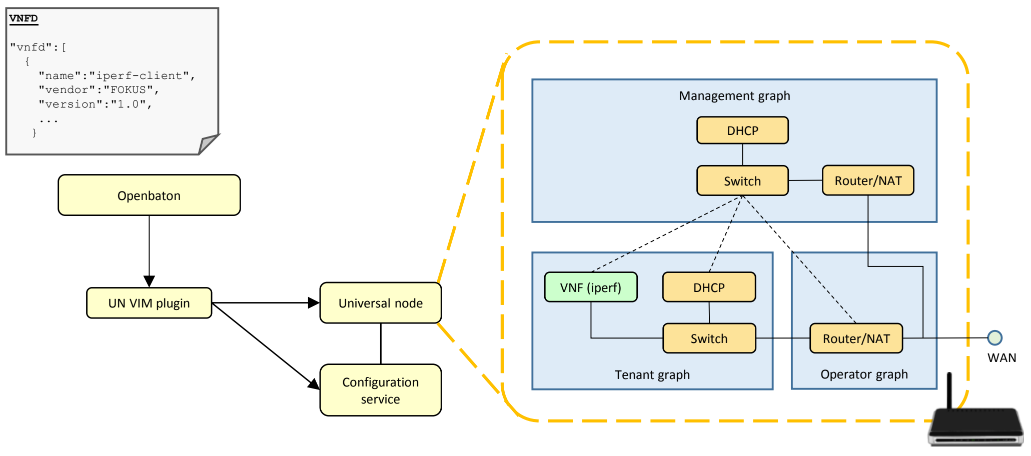
The actual workflow is depicted in Figure 52. First, the UN VIM plugin queries the UN in order to retrieve both the tenant and the management graphs, which are then modified by creating a new VNF in the tenant graph which and by connecting it to both to the management network and to the LAN switches of each network where the VNF has to be attached to. After sending the new updated graphs to the UN, the UN VIM plugin has to retrieve the IP addresses obtained by all the ports of the VNF, which have been be assigned by the DHCP servers serving the subnets where the VNF is attached to. Since the amount of time required to start the VNF and obtain the addresses from the DHCP is unknown, the UN VIM plugins keeps querying all the involved DHCP servers until it finds all the IP addresses it needs, which are identified because the VIM plugin knows the MAC addresses of all the interfaces of the VNF under consideration.

Once the plugin knows all the IP addresses of the VNF interfaces, it checks if some vNICs are associated to a "floating IP". In that case, the UN VIM plugin generates the appropriate couple of *natting* rules (inbound / outbound) and sends them to the Router/NAT in the operator graph by means of the configuration service. Finally, the plugin returns to OB an object representing the VNF just instantiated.



Openbaton
UnPlugin
ConfigurationServer
UniversalNode
createServer(vimInstance,server,list<network>,Map<network,floatingIp>)
getGraphs(vimInstance.Tenant,management)
return userGraph, managementGraph
createVnfServer(userGraph,server.Id)
return vnfServer, userGraph
getVnfSwitch(managementGraph)
return vnfSwitch
connectVnfsOfDifferentGraphs(userGraph,vnfSwitch,managementGraph,vnfServer)
return userGraph, managementGraph
Loop
[For each network in list<network>]
getVnfSwitch(userGraph,network.Id)
return vnfSwitch
connectVnfs(userGraph,vnfSwitch,vnfServer)
return userGraph
updateGraph(userGraph)
getVnfsDhcpInNets(userGraph,list<network>)
return list<vnfDhcp>
Loop
[Until Got All Interfaces Addresses]
Loop
[For each vnfDhcp in list<vnfDhcp>]
getConnectedVnfSwitch(userGraph,vnfDhcp.Id)
return vnfSwitch
getStatus(vnfDhcp.id)
return dhcpStatus
Loop
[For each serverPort in vnfServer.Ports]
If
[areConnected(userGraph,serverPort,vnfSwitch)]
server.Interface.IpAddress=dhcpStatus.getAssociatedIP(serverPort.macAddress)
getRouter(managementGraph)
return router
Loop
[For each network in Map<network,floatingIp>.Keys]
getIpInNetwork(server,network)
return privateIp
configureFloatingRoute(router,privateIp,Map<network,floatingIp>.Value(network))
return server

Figure 52. Server creation: actual workflow.


VNFD

```
"vnfd":[
  {
    "name":"iperf-client",
    "vendor":"FOKUS",
    "version":"1.0",
    ...
  }
```

Figure 53. Deploying a server VNF (in green) and total number of VNFs running (in orange) that are required to emulate the requested service.

As a conclusion, Figure 53 shows the amount of VNF that are deployed in the physical infrastructure, differentiating between the ones under the control of OB (i.e., the green one) and the ones that have been created to emulate the requested service that we can call “service VNFs” (in orange), which are totally unknown to OB. In addition, some connections are created between service VNFs and the management network in order to be able to configure automatically their parameters, driven by the UN VIM plugin.

5.3.3.1.6. The NFV@EDGE generic VNFM

Among the other duties, the VNFM component has to return specific data for (*i*) the VNF to start and configure basic parameters (e.g., turn network interfaces up, download the Element Management System (EMS) software), and (*ii*) additional commands in order to configure the VNF image with the proper software (e.g., download the specific network function, etc.).

While the most part of the above mentioned data does not change whether the VNF runs in an OpenStack or in a UN domain, some minor information have to be adapted. For instance, one of the major changes in this respect is to configure the basic network paths so that the initial downloads are handled through the management network (i.e., the specific vNICs and the associated IP routes), instead of the main data path, which is the default choice for OB.

Therefore the source code of the generic VNFM does not have to be modified at all for the NF@EDGE experiment[6]. However, since the current VNFM instance, running on the SoftFIRE testbed, returns the standard data to the VNF, we hat to create another VNFM instance that was used only in our experiment and that returns the initial data according to the specific setup of the UN.

The second modification refers to the assignment of IP addresses. In fact, when OB asks to the VIM plugin the instantiation of a VNF, it expects the VIM to assign an IP addresses on all the interfaces of the VNF. While Openstack is able to generate such information before the

[6] In fact, this is true except for a minor modification: since the UN does not implement the concept of resource quota, the VNFM cannot update that information with the consumed resources (CPU cores, RAM memory, and available instances); hence, the corresponding call issued by OB to the VNFM fails. Since implementing the quota concept in the UN is far beyond the possibilities of the experiment, we opted for a minor modification of the VNFM code so that a meaningful number is returned instead.

creation of the VNF itself (hence can return it immediately) although the actual setting is postponed for the deployment phase, the UN relies on a DHCP for such a task; hence it is necessary that the user data executes a dhclient on each VNF interface in order to get the address, which is then returned to OB.

5.3.3.2. Integration of the UN in the SoftFIRE testbed

This activity focused on the integration of the UN in the SoftFIRE testbed. This required (*i*) to update the master installation of the OB orchestrator with the new VIM plugin developed in the previous activity, and (*ii*) to integrate a UN in the SoftFIRE testbed in order to run the experiments.

The UN is physically located in the Network and Multimedia Lab @ Politecnico di Torino (POLITO), connected to the softFIRE testbed by means of a GRE over IPsec VPN connection. The SoftFIRE project allocated to the POLITO testbed an entire /24 network, which has been reserved to the experiment in order to have enough IP addresses to assign to the VNFs.

The physical setup of the experiment is depicted in Figure 54, which shows the main location of all the machines and services involved; used services were installed either in Torino, Italy, or in Berlin, Germany, which is enough to emulate a service that benefits from the "NFV@EDGE" approach.

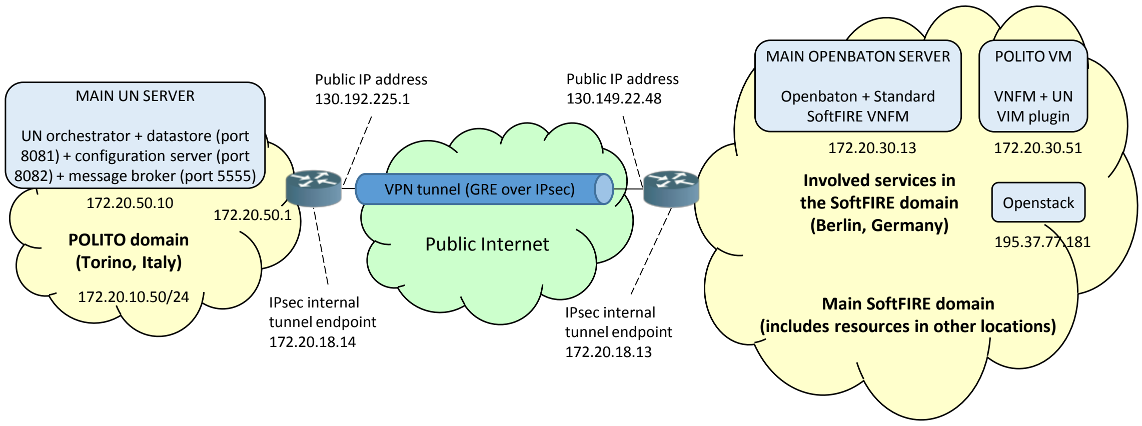

MAIN UN SERVER
UN orchestrator + datastore (port 8081) + configuration server (port 8082) + message broker (port 5555)
172.20.50.10
172.20.50.1
POLITO domain (Torino, Italy)
172.20.10.50/24
Public IP address 130.192.225.1
IPsec internal tunnel endpoint 172.20.18.14
VPN tunnel (GRE over IPsec)
Public Internet
Public IP address 130.149.22.48
IPsec internal tunnel endpoint 172.20.18.13
MAIN OPENBATON SERVER
Openbaton + Standard SoftFIRE VNFM
172.20.30.13
POLITO VM
VNFM + UN VIM plugin
172.20.30.51
Involved services in the SoftFIRE domain (Berlin, Germany)
Openstack
195.37.77.181
Main SoftFIRE domain (includes resources in other locations)

Figure 54. Actual deployment of the NFV@EDGE experiment.

In fact, the physical setup of the experiment was much more complex than what has been described in this document, as it involved many changes (mainly administrative) on the OB side due to the fact that our experiment required a deep modification of the software infrastructure, hence permissions and security policies have been a big issue. However, this large amount of work was carried out by the members of the SoftFIRE project, outside our control. Hence, we cannot report exactly what has been done to allow a new VIM plugin to be integrated in SoftFIRE and still maintain the security and isolation characteristics that are expected in a distributed testbed with several concurrent experiments.

5.3.3.3. Creation of the required VNFs running on the SoftFIRE infrastructure

VNFs are not simple virtual machines with some network functions running in it, but they need to be extended with the proper software in order to be controlled by the orchestrator/VNFM. This activity will take care of creating the proper VMs that are needed to deploy the service that will be used in the validation phase.

The VNFs to be used in the experiment are created starting by a clean disk image that includes only the Ubuntu 14.04LTS operating system with the basic software suggested for cloud deployment, hence a *cloud-init* instance that is used to push arbitrary software customization (e.g., install additional software modules) at run-time in the VNF[7].

In order to work properly, cloud-init needs a *userdata*, which contains the script that is executed when a VNF starts (and that, in our case, configures basic parameters such as VNF ports, and installs the EMS), and a *metadata*, which contains some VNF parameters (such as the VNF *hostname*) that is used by cloud-init for the initial configuration.

There are several ways to push *metadata* and *userdata* to cloud-init; in our case, we included the information in the NFFG, which tells the UN to mount a specific additional data volume in the VNF, where the two files are created. In any case, this is implementation-specific and the specific details are hidden by the VIM plugin in use; for instance, in OpenStack, VNF have to contact a metadata server at boot.

Each VNFM provides the same *userdata* to all the VNFs, which installs the EMS agent that communicates with VNFM through the RabbitMQ system, in our case through the management network. Hence, the *userdata* contains an host-specific IP route that forces all the traffic going to the VNFM to be send through the VNF management interface. After the EMS is installed, it registers to the message bus, hence notifying its presence to the VNFM, and wait for the VNFM to give it some instructions, as shown in the workflow depicted in Figure 55.

In particular, the VNFM tells the agent to clone the repository containing all the scripts associated to the lifecycle of the VNF and, after that, when a lifecycle event is triggered, the VNFM send to the EMS a message specifying which script to execute and the parameters to use.

The lifecycle events we manage in the VNFs used for the tests are: INSTANTIATE, CONFIGURE and START.

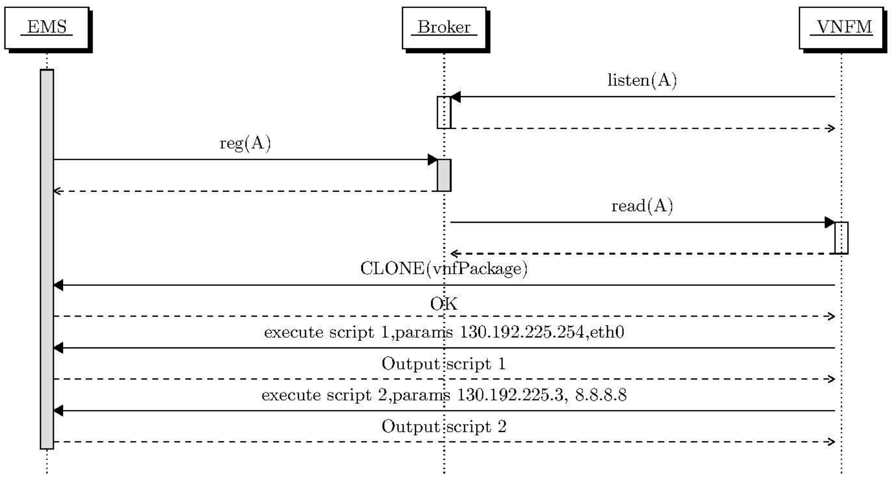

EMS
Broker
VNFM
listen(A)
reg(A)
read(A)
CLONE(vnfPackage)
OK
execute script 1,params 130.192.225.254,eth0
Output script 1
execute script 2,params 130.192.225.3, 8.8.8.8
Output script 2

Figure 55. Registering a VNF in the VNFM: workflow.

[7] The image has been downloaded from https://cloud-images.ubuntu.com/trusty/current/.

5.3.3.4. Validation results

This section presents the results achieved by the NFV@EDGE experiment, distinguishing between the achievements in terms of mapping of the OB software architecture, an initial experimental validation of the benefits of the NFV@EDGE approach.

5.3.3.4.1. Mapping an ETSI MANO service on the UN platform

As far as the results planned in this experiment are concerned, the UN platform has been able to emulate all the requested scenarios and it proved to be very flexible. In fact some limitation have been found when setting up graphs that are related to the overarching OB platform that, due to the complexity of the ETSI MANO architecture, does not allow (yet) to create services that, instead, can be supported by the UN. The most important limitations will be reported in Section 5.3.4.

Although, apparently, the number of VNFs that are required for even a basic service in the UN is very high (e.g., in Figure 53 shows that *six* additional VNFs are required in order to create a minimal service that includes a single VNF connected to a LAN), this may not be troubling, because:

1. The above VNFs may be deployed anyway in the other infrastructure controllers (e.g., OpenStack), which however do not show them to the requesting tenant. In fact, OpenStack implements some default services with Linux containers, which are not that heavy such as a VM, but are still consuming resources.
2. The choice, of the UN, to show exactly how the service is implement, leaves to the service requester the possibility to have the complete control on the service that is actually running. Therefore, the user can decide e.g., whether to start a router or not, can choose the type of router, it can configure any parameter, and more. Default functions, such as in OpenStack, may be good for a basic service, but it may not be appropriate for a complex service as the user cannot have full control on them.
3. Several service VNFs are devoted to the management task, which has been chosen to adopt an out of band model. However, given what has been said before, the management model can be changed at any time, with minimal changes in the UN VIM plugin, hence avoiding the deployment of several VNFs. This confirms (i) that the UN service model, based on graphs, is very powerful, and (ii) that the number of additional VNF can be reduced by simply changing the mapping model between the ETSI MANO service and the physical UN infrastructure.

5.3.3.4.2. Experimental validation

The software architecture described in the previous section has been validated with the testbed depicted in Figure 54, according to the two scenarios presented in Section 5.3.2.1.4 - "Validation of the "NFV at the edge" approach - (Figure 44).

5.3.3.4.2.1 First scenario: remote VPN access

The first scenario consists in offloading the VPN client to the network (e.g., his residential gateway), hence allowing users to connect to their corporate VPN server without having to install anything on their user devices. This permits to get access to the VPN from any device, e.g., also from a smart TV.

In this case, the service is composed of a firewall, a router/NAT and a couple of VNFs playing the role of IPsec endpoints. The service is split between the UN and OpenStack; the firewall and the IPsec client are in the first domain while the router/NAT and the IPsec server are in the second one.

During the tests we measured the time needed to deploy the service, with the corresponding breakdown in the different composing components (e.g., create network, create subnet, create VNF), under the UN. We were not able to perform the same measurement for the portion of the service that has been deployed on the OpenStack domain, as we do not have the control on that domain, hence we cannot introduce the necessary measurement probes. However we noticed that the final deployment time is very similar in the two domains, as the biggest part of the time is spent in setting up the proper VNF image, as shown in Table 9 in the lines "Create VNF".

Interesting, although all the actions mentioned before require starting a VNF (a switch VNF in case of a "create network", a DHCP VNF in case of a "create subnet"), the "create VNF" command is by far the most expensive one. This is due to the fact that the service VNFs deployed by the UN are already fully customized, hence the UN needs only to transfer the image from the image server (i.e., the datastore) and then properly configure the software inside the VNF. Instead, when a VNF is created by OB, the image is a "vanilla" image, hence we need to install the EMS software (through the proper set of *apt-get* commands), then register to the VNFM, and finally download (and configure) the additional software that actually installs the desired network service (e.g., the *firewall* or the *IPsec client* in our use case). Given that the above commands require downloading a possible huge amount of data from the Internet, this explains the much higher deployment time of the "create VNF" command.

Table 9. Service deployment time.

Action	Deployment time (s)
Create network (client – firewall)	3
Create subnet (client – firewall)	10
Create network (firewall – IPsec client)	3
Create subnet (firewall – IPsec client)	9
Create network (IPsec client – external)	3
Create subnet (IPsec client – external)	4
Create VNF (firewall)	50
Create VNF (IPsec client)	43
TOTAL	86 (1 mins, 26 secs)[8]

[8] As detailed in the following, the most part of the VNF deployment occurs in parallel.

In a nutshell, the above actions correspond to the following operations:

- Create network: retrieve the already deployed graphs from UN, creation of a switch, connection of the switch with the tenant NAT.
- Create subnet: retrieve the already deployed graphs from UN, creation of a DHCP VNF, connection of the DHCP to the LAN switches (one vNIC in the tenant graph, the other in the management graph), DHCP configuration.
- VNF deployment: retrieve the already deployed graphs from UN, creation of the VNF, connection of the VNF to each LAN switch of the network it is attached to, execute a dhclient for each VNF interface and wait for a DHCP answer, configuration of the tenant NAT with an IP address for each VNF interface associated to a floating IP.

The overall service deployment time is smaller than the sum of all the deployed entity: in fact, while both the "create network" and "create subnet" are sequential, most of the deployment steps required to start new VNFs occur in parallel. In fact, only a small portion of the total VNF deployment time (about 4 seconds) requires a locking mechanism that forces that portion of the code to be executed sequentially. Hence, the total deployment time of a service is calculated by means of the following formula:

$$\sum t_{network} + \sum t_{subnet} + (n-1) * 4 + max(t_{VNF})$$

where *n* is the number of VNF described by the NSD.

5.3.3.4.2.2 Second scenario: domestic access to the Internet

The second scenario assumes to setup a domestic Internet access (i.e., a LAN switch, DHCP, storage server, NAT, firewall), although the VNFs are installed in different portions of the infrastructure (edge vs cloud), in order to quantify the advantages of the NFV@EDGE scenario. In fact, in this scenario we measure the difference in terms of throughput when contacting the storage service in the three operating conditions presented in Figure 44 (Section 5.3.2.1.4), with the aim at assessing how the user experience improves when the service gets closer to the end user.

First we measured the throughput when the service is entirely deployed into OpenStack, hence using the UN as a (dumb) vCPE; then we moved the storage into the UN; finally we verified how the performance changes if we deploy everything in a single UN.

We repeated the measurement three times, using the *iperf* tool configured to generate 1GB of traffic between the client and the storage server (Table 10).

Table 10. Measuring the throughput in accessing to the storage server.

	Bandwith (Mb/s), measure 1	Bandwith (Mb/s), measure 2	Bandwith (Mb/s), measure 3
Case 1: All services on OpenStack	23.9	21.4	24.3
Case 2: Storage service, DHCP and LAN switch on the UN, the rest on OpenStack	206	210	203
Case 3: All services on the UN	112	118	116

The tests confirm the advantage of the NFV@EDGE approach, as migrating at least some selected VNFs near the end-user improves its network experience. In this case we can show an improvement of approximately one order of magnitude when the storage service is local to the user (hence, both located in Torino, Italy) compared to the case in which the storage server is moved to Berlin, Germany.

It is worth to note that the performance demotes when you deploy the entire service on the UN, such is an expected result as the UN controls next generation home gateways, which usually has no many compute capabilities because it is a low cost device.

5.3.3.5. Analysis of the results based on KPIs

Both the KPIs defined by the experiment and reported in Section 5.3.2.2 have been achieved. In particular, the first KPI has been achieved approximately 9 weeks after starting the project[9] (on Feb 1st), while the second has been achieved at week 12 (on Feb 20th).

KPI-1: UN integrated with SoftFIRE and running at POLITO	
Description	**Result and date**
Two Universal Nodes installed at POLITO premises are up and running and integrated with the SoftFIRE infrastructure.	**Passed.** **Feb 1st, 2017**

KPI-2: NFV@EDGE guarantees better performance than pure cloud service delivery	
Description	**Result and date**
The setup of a service installed partly on the edge device (UN) and partly in the datacenter shows that the NFV@EDGE approach can guarantee better performance than a fully datacenter-based service delivery.	**Shown in Section 5.3.3.4.2.** **Feb 20th, 2017**

[9] The NFV@EDGE started with the official signature at POLITO, dated Dec 2nd.

In fact, a note is needed for KPI-1, as the initial requirement was to carry on the experiments using the *FITEagle* interface. However, due to the poor functioning of the software needed to exploit the above interface, we have been suggested by SoftFIRE members to use directly the OB interface, instead of FITEagle.

This has no impact on our experiment, given that our service is about NFV and that OB is perhaps the best tool available to play with the above technology. In fact, the use of the OB interfaces added more flexibility, as we were able to control our experiment better by tuning our parameters and controlling the status of the service through the GUI.

Finally, the milestone, defined at week 9, and referring at KPI-1 has been reached with the planned schedule.

5.3.4. Lessons learnt

- Satisfaction with the achieved results
- issue/limitation encountered that have been solved or still pending
- improvement possibilities, suggestions/considerations.

NFV@EDGE will provide feedback to the SoftFIRE consortium with respect to the following directions.

- **SoftFIRE architecture**. The integration of an additional technology will help to validate the SoftFIRE architecture, determining how easy is to add a new platform to the existing software/hardware framework. This aspect is important as we expect that future testbeds will need to integrate different technologies not foreseen when the first nucleus of the testbed has been designed. Therefore, the capability to seamless integrate new technologies will be one of the key element for the SoftFIRE platform to continue its ride beyond the lifespan of the project.
- **SoftFIRE documentation**. The NFV@EDGE project will validate the completeness of the documentation that presents how to extend the SoftFIRE architecture and how to deploy an additional testbed/site connected to SoftFIRE. An high-quality documentation is one of the keys that will allow the SoftFIRE testbed to evolve, facilitating the integration of new technologies (hence, possibly new partners) in the testbed. Feedback will be provided with respect to the *completeness* of the documentation, the *organization* of the content, and the presence of *simple examples* that will help possible experimenters to play with the testbed. In this respect, NFV@EDGE will also evaluate the opportunity to deliver some information (e.g., examples) though online videos instead of through the more "traditional" written text.
- **NFV at the edge of the network**. Although this paradigm seems to be very important for future distributed services, a careful evaluation of the trade-offs with respect to pure cloud-based services has still to be done. The initial experimental validation of "edge services" proposed by NFV@EDGE will help the SoftFIRE project to analyze the opportunity to extend this paradigm even more, e.g., by introducing new technologies/testbeds whose goals include also operations at the edge of the network.

5.3.5. Summary and Conclusion

The NFV@EDGE experiment extended the current NFV infrastructure available in SoftFIRE with the capability to control resource-constrained devices, such as home/SOHO gateways, which are very common at the edge of the network.

This has been achieved by integrating the Universal Node (UN), a compact service orchestrator targeted to resource-limited devices, in the OpenBaton toolkit. The experiment (*i*) proves that the integration is possible, hence confirming the flexibility of both the OpenBaton and Universal Node platforms, (*ii*) demonstrates the advantages, in terms of throughput and latency, of deploying edge-based NFV services in specific use cases, (*iii*) confirms the appropriateness of the SoftFIRE testbed that was used to derive the above mentioned measurements in a real, geographically distributed environment. Finally, (*iv*) it shows that the NFV technology is not yet mature for immediate deployment, as confirmed by the several (mostly manual) configuration customizations that are required to make running a service that spans across multiple infrastructure domains, such as an edge-based CPE and a cloud-based OpenStack datacenter, and that can represent the starting point for a possible future work.

5.4 Experiment SECGENE – UniNis (Mentor: DT)

5.4.1. Experiment Description

A consequence of the increasing complexity of federated testbeds for 5G applications is the demand for reducing time to run experiments. Emergent networking technologies, such as SDN and NFV, together with the diversity of radio access technologies (such as LTE, Bluetooth, and Wi-Fi), and the growing trends requiring their simultaneous use, significantly increase learning curve for wireless networking experiments. A promising approach to the problem is automatic generation of target-specific code directly from a high-level experiment description.

The SECGENE experiment addresses the open call's objective related to testbed enhancements in terms of orchestration, control or virtualization capabilities and their real world evaluation over the federated infrastructure by testbed enhancements in terms of development of automatic code generator for experiments.

SEmantics driven Code GENEration for 5G networking experimentation (SECGENE) builds upon the SoftFIRE platform to assist experimentators by generating automatically software code for experiments from a high-level specification. SECGENE takes RSpec definition of an experiment topology, as created by the SoftFIRE platform, augments it with a user defined semantic description of the experiment and generates software code that is directly deployable and executable on the testbed federation. A new ontology will be developed for the semantic representation of the RSpec definitions while an existing ontology framework will be adopted for the semantic annotation of the experiment. Finally, all developed ontologies would be integrated with the updated framework.

With a goal to validate the approach, the experiment proposes that it will collect wireless channels transfer rates data, process the results for knowledge generation, store the knowledge on a server and use the knowledge to reason and make informative decisions about wireless channel usage coordination. As an additional benefit, SECGENE aims to experimentally investigate the performance benefits and unique challenges that the dynamic coordination results in, especially focusing on coexistence in complex federated infrastructures.

5.4.2. Key Performance Indicators

For this experiment 3 KPIs have been identified, which are presented more in detail below:

KPI-1: Development and deployment of NFs

Description: Development and deployment of NFs that provide evidence of the capability to instantiate the SecGene solution manually on SoftFIRE platform.

Type: Boolean (YES or NO)

Means of verification: This KPI will be verified by instantiating within the SoftFIRE infrastructure NFs that will be used to show the ability of running SecGene.

Evaluation result: YES, achieved.

KPI-2: Documentation

Description: Generation of documentation that describes the system and how users can use the SocGene solutions.

Type: Quantitative.

Means of verification: Documentation availability and reviewed by the SoftFIRE project.

Evaluation result: 100% achieved; very accurate and detailed documentation of the SecGene experiments has been made available according to the given schedule.

KPI-3: Demonstration and preparation of movie schowing the experiment

Description: Running of a demo of the system and creating an experiment movie that shows how code will be automatically generated and executed.

Type: Quantitative.

Means of verification: Proof that the demo is running and creation of a film showing this.

Evaluation result: 100% achieved; experiment video and showcase have been produced and can be found at: http://infosys1.elfak.ni.ac.rs/secgene/videos/.

5.4.3. Deliverables

The SecGene experiment has produced a number of deliverables, which are listed below:

D.1.1 Detailed Experiment Design:

This report gives a description of ontologies and automatic code generator that needed to be developed as the basis for executing the proposed experiments. The deliverable also presents details about the design of prototype framework deployments and presents the feedback acquired from the initial testing.

This report gives a status overview of the SECGENE controller implementation and its deployment for conducting experiments. Intermediate experimentation results, the possible problems that have been encountered and the remedies proposed to solve or circumvent the problems are also reported.

D.2.1 Experimental Tutorial:

This document provides the detailed information about the experiment. Based on the experiment setup, this report gives a user manual with all important details and instructions required for reproducing the proposed experiment in testbed environment.

D.3.1 Final report:

This report gives a detailed description of the experiment results, advanced experimentation functionalities offered through the developed experimentation framework, and conclusions made in the process.

D.3.2 Feedback report on the platform:

This report gives feedback on the SoftFIRE platform and the SoftFIRE federation.

D.4.1 Experiment showcase (Movie):

This deliverable is an experiment showcase which has been made available as a movie and can be found at: http://infosys1.elfak.ni.ac.rs/secgene/videos/. It shows the experiment execution over the SoftFIRE infrastructure is available for inclusion on the SoftFIRE web portal and in other networking events where SoftFIRE will be participating.

D.5.1 Promotion and Dissemination report:

This deliverable provides an overview of the dissemination activities during the duration of the experiment. The document collects all the publications and dissemination activities that were achieved by University of Nis Faculty of Electronic Engineering. Dissemination activities that are related to scientific publications and participation in international conferences and events and finally promotion activities are included in this deliverable.

SecGene Technical report:

This document gives an explanation of work carried out and an overview of SecGENE accomplishments. It describes objectives and actual progress based on measurements, the SecGENE management aspects and its practical implementation according to the proposal's Methodology and associated work plan. In the report, the obtain results are described and major achievements are highlighted. This report, which was produced at the end of the experments, includes all other deliverables as annexe and thus is a comprehensive and very detailled summary of all the work carried out.

5.4.4. Lessons learnt

Deliverable 3.2 describes the list of identified significant and useful feedbacks and functionalities on the SoftFIRE platform usage in SecGENE experimenting process. This list is a compilation of the feedback given by implementers of SecGene based on a questionnaire. The main results can be summarized as follows:

- Excellent documentation provided by SoftFIRE.
- Some days are needed to get familiar with the SoftFIRE testbed structure and usage.
- Minor problems were quickly fixed by the SoftFIRE team, which was always responsive and willing to help.
- At the very beginning of the experiment there were some issues but the SoftFIRE platform improved over time.
- The SoftFIRE platform sometimes ran out of resources due to a large number of parallel experimenters.

5.4.5. Summary and Conclusion

The SecGene experiments have successfully finished all the work and objectives according to the plan and absolutely in time. The SecGene team was very committed and able to produce excellent deliverables, despite of some technical problems at the beginning of the test phase. They were extremely positive about the test infrastructure provided by SoftFIRE and about the support. All in all, the collaboration between SecGene and SoftFIRE was more than fruitful for everybody.

5.5 Experiment SOLID – GridNet (Mentor: TIM/U Surrey)

The main objective of SOLID is to leverage the technologies of SDN and NFV provided by the SoftFIRE facilities in order to build a sophisticated offloading framework for heterogeneous networks (LTE/LTE-A & Wi-Fi) that is driven by the end-user perceived QoS.

All required functionality will be built as new Network Functions (NFs) that are fully interoperable with the SoftFIRE framework and will also be left to the community towards realizing new experiments by SoftFIRE users. SDN functionalities will help in implementing seamless offloading maintaining IP connectivity. All the needed mechanisms will be integrated into a practical system implementation that will be evaluated under specifically designed scenarios to showcase the offered advantages.

The framework will be evaluated through a set of experiments with two main scenarios:

1. Initially, small experiments will be performed in order to assess the functionality of the developed NFs and the offloading mechanism. Once the framework is ready for large-scale experimentation, a scenario with multiple users and SLAs will be conducted. This will involve a set of user flows that will utilize the LTE network simultaneously each one with a guaranteed SLA by the provider. Once the LTE network capacity will not be in position to serve all the users by not violating the SLAs, the offloading mechanism will decide which user and traffic flow will be offloaded to the Wi-Fi network. The offloading mechanism should detect in time the possible SLA violation and should undertake the necessary decisions to offload traffic, basing on the output of the dedicated NF, which estimates the network performance of the LTE and Wi-Fi networks.
2. A secondary experimentation scenario will involve traffic flows of different applications, in order to evaluate the offloading mechanism in conjunction with the flow classification NF. The flows will include voice and video traffic, web-browsing, file-transfer and more. The proposed framework should serve voice and video over LTE since it is more stable than Wi-Fi in terms of delay and jitter, whereas large volume traffic could be easily offloaded to the alternative Wi-Fi network when LTE is congested.

This experiment was initially assigned to TIM for mentoring but had to be shifted to UoS as the experimenters needed special equipment, which could be provided only by UoS on-site. The experimenters spent one week at the University fulltime supported by the UoS team for being able to execute their experiment successfully.

5.5.1. SOLID Technical Overview

Figure 56 below illustrates the architecture of the SOLID experiment. The EPC Bridge is a virtualised software component at University of Surrey's OpenStack server. This instance runs on a Virtual Machine (VM) and communicates with virtualised LTE components, i.e. Control

Plane Node (CPN) and User Plane Node (UPN). An SDN controller, Trema, is installed in the VM, and programs the EPC bridge so as to alter traffic from mobile clients, enabling WiFi offloading.

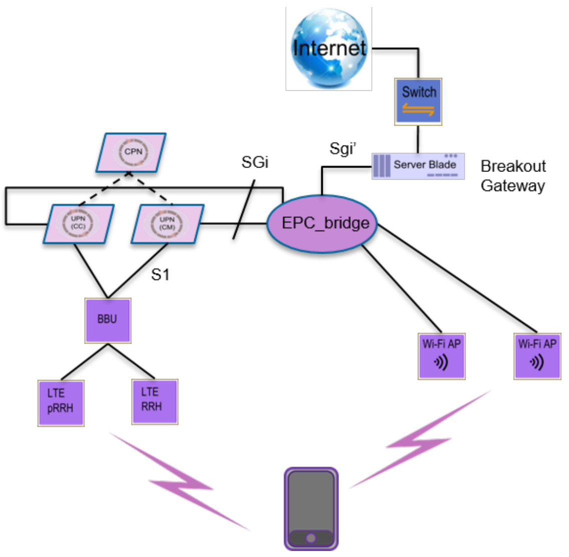


Figure 56. Architecture of the SOLID experiment at UoS LTE and WiFi infrastructure.

Summary of technical details of the SOLID experiment are as follows:

- WiFi mesh network is setup using 802.11s extensions,
- The test setup consists of four nodes with LTE USB dongles, and WiFi cards,
- One node is used as the WiFi Access Gateway (WAG). The WAG is using a point-to-point tunnel with the EPC for relaying all the client traffic,
- On the EPC, we bridge the tunnel end-points of the LTE and WiFi network with the Internet, and manage it through an SDN controller (Trema controller). These are the virtual components deployed at the UoS testbed,
- A control manager application runs on the client nodes
    - Each time a client enters the network, it sends a request with its client ID and the SLA to be met to the network controller running on the EPC server.
- Mobility is preserved by using the same IP addressing scheme during and after offloading.

5.5.2. Experiment Description

SOLID is an experiment that aims to demonstrate virtualized Intelligent Multi-Access User Bearer Control.

The multiple access technologies that are intended to be used in the experiment are Wi-Fi and 4G, LTE-A, although in practice others could also be used.

Many experimental approaches have been tried to affect the goal of Multi-Access User Bearer Control, such as MP-TCP, SIPTO and LWA. However, all of these techniques have significant drawbacks or are a generation away as they require significant modification, as follows:

MP-TCP:	is generally outlawed by operators due to its inherent security risks in passing through firewalls
SIPTO:	has largely been neglected due to its lack of support for Lawful Intercept (LI) and accounting
LWA:	whilst likely to become a staple method for multi-access control, this approach will take several years to become de-facto as it requires significant upgrade of Wi-Fi and LTE base stations in the field as the technique requires modification of the MAC level.

The SOLID experiment aims to provide a demonstration of virtualized Intelligent Multi-Access User Bearer Control by operating a tunneling protocol TUP between User Equipment(s) (UE) over both LTE and Wi-Fi towards a new entity called an EPC-Bridge that can make intelligent routing decisions in coordination with each UE in order to optimize usage of the dynamic performance of each available Radio Bearer (Wi-Fi and/or LTE-A). The EPC-Bridge function is virtualized and connected on the north side of the LTE PGW but before the internet.

A secondary virtualized function is also deployed which is a Controller for the EPC-Bridge.

5.5.3. Key Performance Indicators

The main objective of SOLID is to leverage the technologies of SDN and NFV provided by the SoftFIRE facilities in order to build a sophisticated offloading framework for heterogeneous networks (LTE/LTE-A & Wi-Fi) that is driven by the end-user perceived QoS.

All the required functionality will be built as new Network Functions (NFs) that are fully interoperable with the SoftFIRE framework and will also be left to the community towards realizing new experiments by SoftFIRE users. SDN functionalities will help in implementing seamless offloading maintaining IP connectivity. All the needed mechanisms will be integrated into a practical system implementation that will be evaluated under specifically designed scenarios to showcase the offered advantages.

The framework will be evaluated through a set of experiments with two main scenarios:

1) Initially, small experiments will be performed in order to assess the functionality of the developed NFs and the offloading mechanism. Once the framework is ready for large-scale experimentation, a scenario with multiple users and SLAs will be conducted. This will involve a set of user flows that will utilize the LTE network simultaneously each one with a guaranteed SLA by the provider. Once the LTE network capacity will not be in position to serve all the users by not violating the SLAs, the offloading mechanism will decide which user and traffic flow will be offloaded to the Wi-Fi network. The offloading mechanism should detect in time the possible SLA violation and should undertake the necessary decisions to offload traffic, basing on the output of the dedicated NF, which estimates the network performance of the LTE and Wi-Fi networks.
2) A secondary experimentation scenario will involve traffic flows of different applications, in order to evaluate the offloading mechanism in conjunction with the flow classification NF. The flows will include voice and video traffic, web-browsing, file-transfer and more. The proposed framework should serve voice and video over LTE since it is more stable than Wi-Fi in terms of delay and jitter, whereas large volume traffic could be easily offloaded to the alternative Wi-Fi network when LTE is congested.

5.5.3.1. *KPI 1: Aggregated System Throughput Increase Due To Offloading*

This KPI focuses on the percentage of aggregated system throughput increase due to offloading; e.g. if the network consists of a single eNodeB with 100Mbps link and a UE equipped with 802.11ac. The solution aims to increase the network capacity by at least 100%.

5.5.3.2. *KPI 2: Percentage Total Network Capacity Increase In Meeting User SLA's*

This KPI focuses on the percentage of total network capacity increase for meeting the user SLAs. In practice this metric will show the number of extra users that could be served thanks to offloading.

5.5.4. Results and Analysis based on KPIs

The following results were extracted during multiple executions of the same experiment scenario, in order to minimize any random fluctuations during the runs and provide meaningful outcomes. More specifically the scenario included 3 users/clients where each of them had a previously agreed SLA concerning the minimum bandwidth of its downlink (DL). The SLAs are the following:

- Client1 requests 30 Mbps
- Client2 requests 35 Mbps
- Client3 requests 20 Mbps

The total required throughput would be 85 Mbps, while the maximum throughput of the Femto Cell using the USB LTE Dongles experienced stable behavior up to 70 Mbps. According to these, in order for our system to achieve the required SLAs, we needed to offload the user with the maximum bandwidth (i.e. 35 Mbps) to the Wi-Fi network and continue serving the other two clients over the LTE network. This has been measured and analyzed in the following table.

The following table presents the KPIs measured by the SOLID experiment:

	Description	**Target Value**	**Measured Value**
KP#1	**Aggregated System Throughput Increase due to Offloading** This KPI focuses on the percentage of aggregated system throughput increase due to offloading; e.g. if the network consists of a single eNodeB with 100Mbps link and a UE equipped with 802.11ac. The solution aims to increase the network capacity by at least 100%.	85 Mbps	82 Mbps
KP#2	**Percentage Total Network Capacity Increase in Meeting User SLAs** This KPI focuses on the percentage of total network capacity increase for meeting the user SLAs. In practice this metric will show the number of extra users that could be served thanks to offloading.	50%	50%

Table 11: KPI Results for SOLID

5.5.5. Experiment Extensions

Removal of TUP from the solution

At present the SOLID system operates using TUP (Telephone User Part) tunnels which enable the user plane traffic to be tunneled to the EPC-Bridge (bridging Wi-Fi and LTE bearers).

However, this protocol overhead adds some inefficiency over and above that introduced by TCP/IP and GTP. The only way around this (to remove the TUP protocol from the solution) would need to do the following:

- integrate the EPC-Bridge into the EPC solution
- operate the EPC-Bridge as a trusted component with Internet access or operate it as a fixed UE proxy.

… this would require further development by UNIS and the SOLID team.

5.5.6. Lessons learnt

In this section, the issues that were encountered during the course of the experiments are outlined, as well as the actions taken to resolve them. We also explain how to make improvements on the presented solution's integration with an existing LTE infrastructure.

5.5.6.1. Issues and Limitation Encountered, and Solutions:

Issue 1: Specific Wi-Fi Service Set Identifier (SSID):

The team has initially concluded that there needed to be a separate SSID for the experimental system on the existing Wi-Fi network at UoS. Later, after careful Wi-Fi configuration, it was possible to manage the experiment without a separate SSID.

Issue 2: Virtualisation System Reset:

There have been issues restarting the Virtualisation system at UoS, as UoS experienced several catastrophic power failures in December 2016, making it difficult to restore the OpenStack [] virtualization infrastructure controller. The open source repositories had been removed and replaced, which made it difficult to set up the prior system, due to multiple software dependencies that OpenStack relied on. However, UoS was able to bring back the virtualization platform, and made it available in time.

Issue 3: Network Address Translation at LTE core software: The LTE software at UoS, which is part proprietary, uses IP NAT at Packet Data Network Gateway (PGW), converting the source IP as the IP of the PGW component. This is not an issue in the 5G core however, as the IP addressing of mobile clients are handled by separate network components designed and developed at UoS. In contrast, for 4G, the NAT operation at the PGW means that the virtualized software, which is north of the PGW cannot address a mobile client directly. As a solution, packets are tunneled between mobile clients and their virtualized software.

Issue 4: Open Baton and OpenStack Latest Release Incompatibility: Due to the fact that UoS had to upgrade their OpenStack system from OpenStack Liberty to OpenStack Newton, and that OpenBaton is currently not upgraded/tested to run with Newton (only Liberty) then the experiment had to be virtualized locally in the testbed in UoS. However, the virtualisation was successful and only the federated part was not demonstrable. This issue is planned to be resolved for Open Call #02.

Issue 5: The mobile clients are Ubuntu laptops provided by UoS as well as nodes provided by GridNet. Connectivity to LTE is via USB dongles. LTE on USB dongles has performance issues, as connectivity may become intermittent at some nodes, requiring manual inspection and reshuffling USB dongles frequently.

Issue 6: The SOLID experiment needs multiple interfaces at each Virtual Machine (VM). The initial test platform in which the experiment software was tested on a fixed LTE did not have such issue, as the software was hosted on a real server. However, OpenStack provides a single interface per VM. To remedy this, UoS found a solution to add multiple internal subnets, each with a separate port to the experiment VM. However, since OpenStack has a single external router, the connectivity to the VM was through only a single Ethernet interface at a time. GridNet modified the software, so that traffic in/out of the VM would trace a single interface at the VM.

Issue 7: The connectivity tests took longer than expected, and spanned more than a week. UoS provided GridNet with remote access capability to the WiFi network, and their VM on OpenStack.

5.5.6.2. Experiment Extension

At present the SOLID system operates using TUP tunnels, which enable the user plane traffic to be tunneled to the EPC-Bridge (bridging Wi-Fi and LTE bearers). However, this protocol overhead adds some inefficiency over and above that introduced by TCP/IP and the GPRS Tunneling Protocol (GTP). The only way around this (to remove the TUP protocol from the solution) would need to do the following:

- integrate the EPC-Bridge into the EPC solution,
- operate the EPC-Bridge as a trusted component with Internet access or operate it as a fixed UE proxy,

This would require further collaborative work between University of Surrey and GridNet.

5.5.7. Recommendations

Recommendation 01: It is recommended that for and experiment of this complexity involving Wi-Fi, LTE Core, and LTE RAN, bridging between and packaging a new VNF, then we need to engage at a more detailed level prior to the experimenter team visiting the site to operate real mobiles on such a system.

The amount of time educating the SOLID visiting team in how to package their VNF into a VNFD has been notable and UNIS plan to produce a document explaining the process in more detail that we can use to educate new Experimented for the next stage in Open Call #02.

UNIS also plan to produce additional documentation on the options for operating efficiently with the team at UNIS in order to test real mobile devices either remotely or locally on the RANs available with the experimenters VNFs.

5.5.8. Summary and Conclusion

The realization of our experiment in a real setup enabled us to verify that our theoretical assumptions regarding the offloading mechanism could be technically achieved in a real LTE network. Even though the duration of the experiment was limited, we managed to extract valuable knowledge regarding the adaptations needed from our mechanism in order to operate in a real LTE network. There are things left for further investigation and experimentation, in order to find different ways of making the offloading mechanism as seamless as possible for the client and the operator.

5.6 Experiment Tracking FLE – Fujitsu (Mentor: U Surrey)

5.6.1. Experiment Description

The FLE experiment was designed to enable a Video Analytics server to operate as an edge application addressed by a mobile so that it can act as a proxy video feed to the User local enhanced video feeds. When the user connects to the edge application server and logs on to the Video analytics application, the User is able to receive enhanced video feeds from cameras in their area in order to better understand the area and the situation in the area and the vicinity of the area. The Video Analytics application is operated at the edge of the RAN so that it can reduce latency in sending video feeds to the User. Also as the edge application is anchored in the network it has a much larger scope access to the area than the User.

The setup in the SoftFIRE testbed is where the Video Analytics Edge application is packaged for deployment into a component tested that has LTE and/ or 5G RAN and then operated at the SGW/PGW level to provide intranet hosting of the application in order to reduce latency and improve user video scope. The application is intended to be deployed as a VNF, NSD into the RAN capable component testbed onto a local OpenStack Compute server, using OpenBaton.

5.6.2. Key Performance Indicators

Ultra-low latency applications, as well support of IoT devices and applications are two targets of the next generation 5G networks. The project work carried out by the experimenter FLE implements a low latency video analytics and camera control application on the SoftFIRE testbed, particularly the 5G testbed located at University of Surrey (UoS).

The application takes input from a number of camera feeds over the radio interface and consumes these within a virtualised video analytics module, which is packaged as a virtual network function. The network function is initiated via the OpenBaton orchestrator on the UoS OpenStack controlled testbed. The module processes the input video streams, makes decisions, and then issues actions, such as control of a camera to track an object. In order to make effective decisions, the application requires adequate and timely delivered measurements from the infrastructure to ensure that its goal is achieved: effective target tracking with minimal latency.

The FLE experiment focusses on the following goals:

1) Determine the processing and service latency. If the targets are not met, then determine the reasons/factors of performance degradation,
2) Determine if resource management approaches can overcome any limiting factors,
3) Evaluate the performance improvement gained by mobile edge computing (MEC) as compared to cloud computing and different management strategies,
4) Evaluate the current implementation of NFV and SDN in the 5G testbed, with respect to meeting the requirements of a MEC-based video analytics application.

The following are the key performance indicators of the FLE's experiments on the 5G testbed:

5.6.2.1. *KPI 1: Throughput*

This is the TCP/IP throughput between a UE and the SGi interface. This is the 5G interface between its Packet Data Network (PDN) Gateway (PGW) and an external PDN. Downlink (DL) throughput target is 60 Mbit/s, and UL throughout target is 20 Mbit/s.

5.6.2.2. *KPI 2: Feed switching time delay*

The application determines when the video feed concerning a particular object of interest should be switched to a more appropriate alternative feed and which auxiliary camera should be selected, instructed and remotely configured in real time to provide the alternative video feed regarding the particular object of interest. This KPI is the latency experienced in switching between different video feeds, with a target of less than 1 second. The latency is measured from the time when the decision to make the switch is made until the switch has been completed and the object is being tracked by a different camera.

5.6.2.3. *KPI 3: Number of processor cores in use*

This is the number of CPU cores that the application uses to process the received video feeds, caused by three factors: input streaming from cameras, output streaming towards cameras, and the video analytics VNF. A low number of cores is desired, with a maximum allowed 14 cores.

5.6.2.4. *KPI 4: Mean CPU utilization*

This is the percentage of utilization of CPUs in use by the application. A maximum of average 70% CPU usage is targeted, across all used CPUs.

5.6.2.5. *KPI 5: Number of camera feeds*

This is the number of camera feeds required to achieve the desired action, i.e. target tracking, under system constraints, which are number of CPUs and CPU utilization.

5.6.3. Results

Unfortunately, this experiment was abandoned in the SoftFIRE Open Call #01 as the Experimenter organization lost the experimenter who was working on the project through a resignation in the company and was unable to replace the engineer in the timescale.

Also the experimenter company FLE was not able to get internal clearance to approve the contract before the cut-off signature date.

However, the experimenter did manage to take their Edge Video Analytics Image and package it up into a VNFD suitable for deployment on the local OpenStack UNIS testbed. The application was tested locally on the UNIS testbed and found to be functionally working correctly. Unfortunately, the package was never deployed via the Open Baton orchestrator and was never tested beyond this stage.

We have been encouraging the experimenter organization 'FLE' to pursue a re-application this year in 2017 to repeat the experiment and seek to take it to completion this time in Open Call #02.

5.6.4. Experiment Extensions

EDGE Server

During the active period of this incomplete experiment, the experimenter and the UNIS staff developed a good rapport and managed to demonstrate edge application functionality by deploying the edge application on the same server as the SGW/PGW compute server in effect demonstrating network edge MEC functionality.

The work done was extremely promising and we aim to work with FLE to complete next open call.

Local Video Stream Support

UNIS and FLE found that the application has difficulty getting out to the internet unless it is registered itself as a User Equipment.

The workaround we recommend to resolve this issue is to develop the Edge Server as if it is a fixed mobile – implying more development work by UNIS that is required to enable this capability with a UE emulator software development exercise.

The other alternative workaround is for the operator to provide the video camera feeds internally to the Operator Internet (or hosted by them as a service) so that the Edge Application can access them directly and securely within the operator intranet network.

5.6.5. Issues and Limitations

Issue 01: Edge Application local breakout – see previous section under Local Video Stream support

5.6.6. Recommendations

Recommendation 01: Allow time for large companies to seek approval for contract signature

Often it is not uncommon for a large company to take 4-6 weeks' time to run their approval process to agree to any kind of collaborative working relationship such as is the case with the SoftFIRE open Call process.

Recommendation 02: Allow time for Experimenters to learn how to package their images

It typically takes between 2-3 weeks for an experimenter to learn, understand and be able to complete a packaging of their software, to a suitable standard for deployment onto the federated. SoftFIRE testbed.

Recommendation 03: FLE should re-submit this experiment for Open-Call #02.

5.6.7. Conclusions

Conclusion 01: This was a promising experiment with validity for 5G.

6 Lessons Learnt

The first wave of experimentation has been concluded on the SoftFIRE Federated Testbed. All experiments concluded their work in the allocated period and provided valuable feedback to the consortium. The experiments were covering a broad range of potential applications and feature of NFV/SDN/5G platforms and this is a strong indication for the future work of the SoftFIRE project. Experimenters (especially those that do not have access to proprietary and closed framework) are eager and in need to access open infrastructure for creating a European ecosystem capable of providing viable and innovative solutions.

The SoftFIRE project recognizes that the platform is in its initial stage (and as such it can show some issues), but, at the same time, it is very convinced to have provided a unique and important (even at worldwide level) infrastructure. The second Open Call is under definition and it will leverage the good points and the structural approach followed for the first one. Some simplifications on the procedures of evaluation and selection will take place in order to streamline the process. The platform is under consolidation and it is extended with further important functionalities that new experiments will beneficially leverage and exploit. The interest created by the first Open Call is a good starting point for further capture the attention of a community strongly interested in working in this promising area. After the first "stress" period that forced the SoftFIRE project to consolidate and make the platform more robust, there is more confidence in the possibility to sustain more experiments.

The processes put in place by the consortium (even if they were time consuming) proved to be valuable to support the experiments. The Mentorship was quite useful and effective in order to sort out the issues. The next Open Calls will surely adopt these mechanisms. One point of attention is that the number of experiments supported by the SoftFIRE Federated Testbed will increase and in perspective they could draw all the available manpower and resources. As said, we consider the SoftFIRE platform as an evolving platform that requires development and incremental evolution. The challenge will be to tradeoff between support to the experimenters and development/consolidation capabilities.

The physical platform has supported the current set of experiments. The capability was enough and the experiments did not pose stringent requirement on the communications between different platform components or the distributed virtual machines. Increasing the number of experiments will also require extending the physical capabilities. For this reason, the SoftFIRE consortium is globally increasing the physical elements allocated to the platform and is also introducing new testbeds in order to be able to guarantee the needed processing and storage requests. The communications side will be further stressed out with the new experiment wave. Relying on the Internet is the current possibility of the project, however if this will create too much issues, the consortium will seek other solutions.

Some experiments have used basic SDN functionalities. Many proposals for experiments were proposing innovative solutions based on large SDN implementation. The assessment of the project for the time being is that the NFV technologies are ahead of SDN technologies. There is however a strong demand for them. The project has decided to extend the individual testbeds in order to support basic and possibly advanced SDN functions in order to create a stimulating environment for the experimenters and to offer the possibility to further show the power of integrated NFV/SDN functions in the context of 5G. Obviously moving along this path introduced more risks and the platform could become more fragile and exposed to issues, problems and inconsistencies.

In any case, the SoftFIRE will take the risk and will push for demonstrating the integration. The consortium deems that the project is on the right path and could give a very valuable European positioning to these new technologies. For this reason, currently the SDN functions are under configuration, screening and consolidation so that they can be offered and used by experimenters.

As seen, the SOftFIRE project has reached a set of remarkable results. However, the experimentations have highlighted some inconsistencies and issues in the Federated Testbed and its supporting processes. A few points are listed following:

- The lack of tools for distributing automatically the load over different testbeds
- The lack of "garbage collection" processes for stopping virtual machines that are not used and guarantee the possibility to free existing resources
- The lack of SDN functionalities in each federated testbed
- The structural fragility of OpenStack and the importance to get a full support for the needs of the telecommunication industry
- The mismatch in this context of FIRE tools (e.g., jFED) and the need to prepare OpenBaton to be fully exposed to experimenters.

These and other issues are under consideration within the project. A first assessment on the platform is that OpenBaton implementation has the needed qualities to be a candidate for further steps towards the wide adoption within Open Source implementation and possibly in industrial one. The SoftFIRE project will support this evolution as much as possible in order to position it as a candidate for orchestration in large projects. This would be a major contribution.

At the process level, it should be stressed out that the Project has devoted a large effort for supporting divergent experiments needs. In order to run experiments, the project spent a considerable amount of time for putting in place the platform and particular resources for specific experiments and for instructing and drive the related teams. Putting in place an infrastructure similar to SoftFIRE is a huge investment and many experiments took the opportunity to ask special usages of it. This issue has two facets:

- Experimenter's lack of understanding on programming of (distributed) Virtualized platforms. Many experimenters did not have the experience behind NFV/SDN platform programming. They took the Open Call as an opportunity to put hands on a distributed platform and use it as a learning tool without affording the cost of a large configuration (see the OpenStack point). This has consumed a lot of SoftFIRE project resources in order to put these groups to productivity.
- Some experimenters asked for new and very specialized (proprietary) functionalities well ahead of the current status of the art. Some experimenters took the opportunity of the Open Call to request to the SoftFIRE testbed functionalities and resources that are very difficult to get and that are missing in almost all existing experimental testbed. This was e.g., a type of request to make available the latest particular technologies in the area of NFV/SDN functionalities for (again) educational purposes. This issue together with the previous one has delayed the operations and development of the platform.

The SoftFIRE project deems the results achieved in this period very encouraging despite of all the needed effort to achieve them. The project did not stop to the assessment level of the technologies, but quite soon moved into contribution to the implementation and solutions of important issues within the NFV/SDN realm. In addition, it is quickly moving towards the provision of interesting resources and functions that allow to experiment services, applications and functions for the 5G network evolution.

Experimenters gave their feedback either positive or for improvements. Please find here reference to experimenters' sections on feedbacks reported into present doc:

- EXPOSE, section 5.1.6
- MARS, section 5.2.5
- NFV@EDGE, section 5.3.4
- SECGENE, section 5.4.4
- SOLID, section 5.5.6
- Tracking FLE, section 5.6.6

7 References

ETSI. (2014). Network Functions Virtualization (NFV); Management and Orchestration.

FITeagle, A semantic resource management framework. (n.d.). Retrieved from http://fiteagle.github.io/

Open Baton, a ETSI NFV compliant Network Function Virtualization Orchestrator. (n.d.). Retrieved from http://openbaton.github.io/

OpenStack, open source software for creating private and public clouds. (n.d.). Retrieved from https://www.openstack.org/

Redmine, a flexible project management web application. (n.d.). Retrieved from http://www.redmine.org/

SoftFIRE, Software Defined Networks and Network Function Virtualization Testbed within FIRE+. (n.d.). Retrieved from http://www.softfire.eu

GridNet, http://gridnet.gr/

Multi-Path TCP (MP-TCP), https://www.multipath-tcp.org/

Selected-IP Traffic Offload (SIPTO), http://blog.3g4g.co.uk/2010/09/selected-ip-traffic-offload-sipto.html

Trema controller, https://trema.github.io/trema/

LTE HetNet (LTE-H) a.k.a. LTE WiFi Link Aggregation, http://blog.3g4g.co.uk/2015/04/lte-hetnet-lte-h-aka-lte-wi-fi-link.html

8 List of Acronyms and Abbreviations

Acronym	Meaning
API	Application Programming Interface
DoS	Denial of Service
EMS	Element Management System
ETSI	European Telecommunications Standards Institute
MANO	Management and Orchestration
NF	Network Function
NFFG	Network Function Forwarding Graph
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
OB	Open Baton
PoP	Point of Presence
QoS	Quality of Service
REST	Representational State Transfer
SFA	Slice-based Federation Architecture
SLA	Service Level Agreement
SSL	Secure Socket Layer
TLS	Transport Layer Security
UE	User Equipment
UN	Universal Node
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network