Software Defined Networks and Network Function Virtualisation Testbed within FIRE+

White Paper

SoftFIRE Approach to Experiment Management: Why and How

Jun 2017

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

Table of Contents

List of Figures

1 Introduction: Why do we need a Framework for Experimentation

The SDN/NFV combination promises to be the basis for a deep transformation in the Telecommunications industry. Virtualization offers the possibility to introduce virtualized function for the execution of provision of network services. This means that the entire cloudified infrastructure of an Operator can be virtualized and extended by means of programmability and APIs. As a complement to this capability, the Software Defined Networking, SDN, offers the unprecedented capability to program connectivity resources. This has the consequence that networks and resources can be created obeying specific goals. The further virtualization of SDN resources creates additional value because the control and programmability of network resources goes hand in hand with the possibility to virtualize the different networks and to provide to final customers new controls and customization of network services. In addition, network functions can be segmented and could create "slices" of functionalities for supporting specific application domains.

The relation between NFV and SDN is quite interesting and the industry is striving for a better integration between the two in order to make them enablers for the 5G infrastructure. In the meantime, the two technologies themselves have to be consolidated in order to reach a level of reliability and efficiency that allows their industrial usage. Their integration is not fully achieved and SDN is mainly supporting the communications need of a single installation of OpenStack. The global interconnectivity and segmentation of APIs to be exposed to different programmers is under development and it is not consolidated yet. SoftFIRE will try to integrate in the future.

In order to assess and usage the NFV/SDN technology, SoftFIRE has put in place a Federated infrastructure that aims at assessing and proving the degree of interworking, programmability and security that the solutions have achieved.

NFV and SDN are naturally prone to require distributed processing capabilities. Even in a single testbed implementation, different interacting machines are used to support the exploitation of the virtualization and control functions. The federation of different distributed systems poses additional issues and challenges in terms of interworking, programmability and security because homogeneous execution and management rules have to be granted over different heterogeneous basic (hardware and software) infrastructures.

Designing, programming, executing and monitoring different (virtualized) distributed components requires an efficient orchestration engine and the capability to easily satisfy needed deployment, monitoring, execution and location requirements of the functionalities.

This makes the task of programming and deploying software on virtualizing interworking platform a complex one. For this reason, some tools for supporting the designers of the software can help in defining the required deployment and execution configuration. The challenge to easily design, deploy, instantiate and monitor the distributed applications over virtualized federated infrastructure is even exacerbated in a context as SoftFIRE. As said, one of the requirements is to support different basic technologies, and in addition, each component testbed may provide functionalities that are very specific to the local hardware infrastructure, or they may provide specific services non-replicable elsewhere. This is a problem in itself (even if the SoftFIRE platform was a single user platform) that is further complicated by the fact that SoftFIRE is multitenant. Different Experiments will run in parallel and will have different and sometimes competing requirements in terms of usage of the platform and its resources. It is important to understand who is requesting which resource and for how long, and once the resource is allocated and ready to execute, it is important to monitor its usage and account for it.

The focus of SoftFIRE is mainly evolution to 5G and its enabling technologies (NFV/SDN essentially), so the requirements for resource descriptions and usage clearly fall in specific domain of future telecommunications.

In order to support these and other functionalities, SoftFIRE needs a set of flexible tools that can be offered to experimenters in order to support them during the life cycle of their experiment. These tools have to differentiate between different users, they have to clearly segment and support a specific execution environment, they have to allow for exposition of interfaces of existing services and for access to a repository of pre-existing functions and services that can be tied together by the experiment in order to create compelling applications. The experimenters should also be free to decide what functions to instantiate and where and when to instantiate them.

A graphical user interface would clearly be an advantage for easing the chaining of functionalities and their distribution over the platform. Moving virtual network function over the infrastructure will be a major advantage for experimenters that could find in this way better configurations for their deployment.

The availability of existing tools able to offer these functionalities into a specific environment tailored for SDN/NFV resources is clearly an advantage and probably an enabler for providing these solution on an industrial level.

2 Why we started with the FIRE framework: FIRE APIs, what are FIRE APIs

In a first phase of its lifecycle, SoftFIRE needed to focus on achieving and guaranteeing the interoperability and interworking of very different testbeds. Much of the work has been devoted to this daunting task. It was clear since the definition phase of the project that relying on existing tools could cut down the development time and would allow the project to focus on relevant tasks. From the “platform” perspective”, the requirements were clearly identifiable. The tools should provide:

- User Authentication

- User Authorization
- Resource discovery
- Experiment definition
- Resource reservation
- Resource provisioning
- Experiment control during execution
- Experiment Monitoring

And the choice of FIT Eagle (ref) and jFED (ref) was straightforward. These tools also guarantee a level of conformance to the entire FIRE action and the potential possibility to ease the usage of other FIRE platforms.

The project started the integration of these tools into the SoftFIRE middleware infrastructure in order to offer and exploit their functionalities to experimenters of the first SoftFIRE Open Call.

The adopted configuration within the SoftFIRE middleware is represented in Figure 1: SoftFIRE initial Architecture (integrated with FITeagle and jFED).

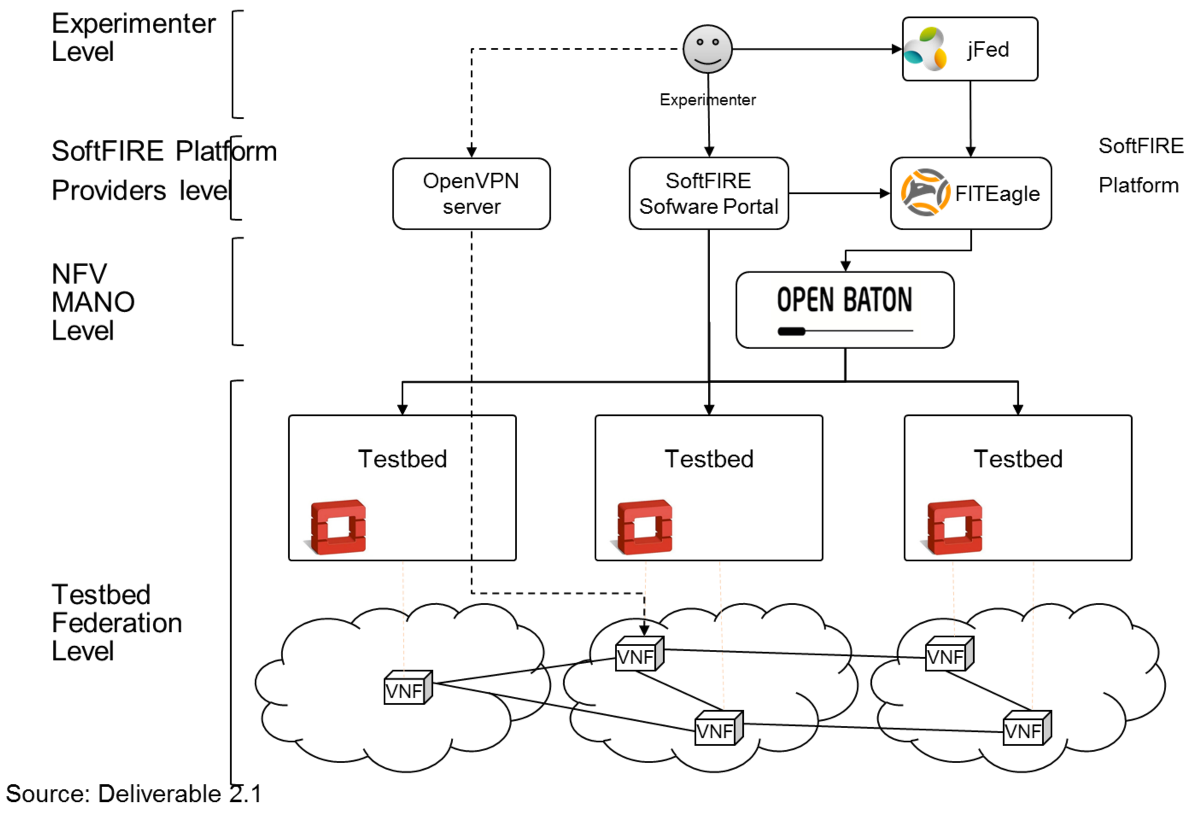

Figure 1: SoftFIRE initial Architecture (integrated with FITeagle and jFED)

From an architectural view point, it can be appreciated that SoftFIRE had to complement the tools with other functionalities such as the Software Portal. This somehow created some idiosyncrasies with respect to the middleware architecture definition. In addition, some overlapping of functionalities was evident due to some replication of resource management between the FIRE tools and the MANO Orchestrator definition.

During the execution of the first Open Call, the tools did not fully maintain their promise for easing the work of experimenters. They showed issues in properly representing and dealing with NFV resource (NFS), while they were properly usable for more basic resources (e.g., VMs). In addition, some of the SoftFIRE experimenters were ready to directly use some of the APIs provided by Open Baton. This has led the project to reconsider the usage of the tools and to develop solutions more readily integrated and usable within a NFV/SDN context.

The new set of functionalities (extended also to support new functions related to monitoring, security, integration of specific physical resources) has been collected under the definition of a new software framework that will leverage and will be fully integrated with the NFV/SDN orchestration. The framework is considered as a valuable contribution to the entire NFV/SDN community (one of the goal of SoftFIRE is to serve as an enabler for the technological evolution of this sector) and possibly a contribution to extend and integrate into FIRE these resources and capabilities.

The goal of this SoftFIRE white paper is to share within the largest community possible the reasoning that have yield the project to endeavor into the development of an Experiment Manager fully devoted to NFV/SDN/5G technologies and the high level design of it. We believe that this discussion is relevant for the NFV/SDN/5G community and we offer our experience and solutions to the discussion and evaluation of other projects or initiatives in order to determine if this is meaningful and viable approach.

3 Limitations encountered

During the First year of the SoftFIRE project a FIRE compliant architecture has been implemented for exposing resources available at each individual testbed.

The introduction of the FIRE APIs has been needed only for maintaining compatibility with the existing FIRE tools, since most of the functionalities required for building up the federation were already present in Open Baton.

Furthermore, providing the experimenters access to the infrastructure only via the FIRE tools limited the set of functionalities compared to the extended set of APIs provided via the Open Baton NFVO. For instance, scaling, one of the most important NFV mechanisms, would require the execution of a set of update functionalities on the existing experiment, while in Open Baton it would only require the execution of a single API. In addition to this, there are use cases that require the instantiation of additional sets of components that cannot be realized without the direct access to the Open Baton APIs. Nevertheless, the Open Baton API does not provide support for all of the Lifecycle events specified by the SFA API that are used by FIRE. These missing features need to be implemented as a thin layer inside the SoftFIRE middleware. Later in this Document a new architecture providing the missing features is introduced.

In SFA resources are described via the Resource Specification (RSpec) language. With those RSpec definitions, resources could become part of a catalogue exposed to experimenters. From the experimenter’s point of view, a SoftFIRE experiment definition is a two-step approach. First, the NS/VNF is defined following the NFV MANO specification, whereas the provisioning of the experiment is then done via RSpec language. Feedback from the Experimenters was asked for a simplification of this process into an better integrated approach using only a single specification language.

3.1 Feedback from SoftFIRE experimenters (call1)

During the first wave of experiments of the SoftFIRE project the experimenters gave a lot of feedback regarding the Usability of the Platform. This feedback concludes that the FIRE approach and especially the SFA API was not flexible enough to be used for the move to NFV paradigms.

One major feedback is the complexity of the Error reporting during the Development of own NFV components. The SoftFIRE multi-layer architecture introduces several integration points which reduce the reliability of the whole system. Furthermore, additional effort is required for adapting the information while passing it from one level to the other. It is important to underline that in SoftFIRE the first level federation is already achieved at the infrastructure level by using OpenStack as Virtual Infrastructure Manager.

Troubleshooting and Supporting the Experimenters during the phase of experiment development caused a lot of effort by both the SoftFIRE team and the Experimenters. Of course this is not an issue if premade NFV packages are used, which were already customized and tested to work with the platform.

As the Experimenters are already using the MANO specifications to develop their NFVs it is more convenient for them to directly utilize the MANO APIs to also control their experiment lifecycle.

3.2 Experiences from other (FIRE) projects

The FIRE initiative included many Projects that contributed to the field of federated Testbeds. Some of them where following the SFA approach for Experiment reservation and control, while others are giving direct access to the individual API of the testbed. However based on our knowledge there are very limited number of commercial SFA-enabled federated testbeds.

BonFIRE[1]

The BonFIRE project does not use SFA, instead is uses a custom API based on OCCI (Open Cloud Computing Interface) to define and control an Experiment on their distributed Testbed infrastructure. The so-called BonFIRE API utilizes REST and XML to define an experiment composed of multiple resources as nodes. The project was very successful in respect of sustainability and was translated into the BonFIRE Foundation, which will continue to operate the BonFIRE multi-site Cloud testing facility.

Fed4FIRE[2]

The Fed4FIRE created a federation containing most of the FIRE based projects across Europe to made the testbed facilities available using a common access point. Some of the federated projects are using SFA based tools to control their experiments. The developers of the jFED tool, iMinds where part of the project, which resulted in good integration and customization of the tool towards the use-cases of the project.

[1] http://www.bonfire-project.eu
[2] https://www.fed4fire.eu

Trescimo[3]

The Trescimo Project provided a federated Testbed environment for M2M communication in the field of Smart Cities in South Africa and Europe. For the Orchestration and Management the predecessor of OpenBaton, OpenSDNCore Orchestrator was used together with FITeage. The architecture already utilized the TOSCA language to describe the Experiments. (1)

OPNFV: Pharos[4]

Besides being not a FIRE project the Pharos Project deals with developing an OPNFV lab infrastructure that is geographically and technically diverse. It provides guidelines and Infrastructure to setup and manage industry-oriented Testbeds with focus on SDN and NFV. Depending on the specific rules for each lab, Experimenters will get direct access to the underlying OpenStack and OpendayLight API using secured OpenVPN access. This approach was very well adapted by the community which is formed by a large number of companies that are providing testbeds to the project.

4 A New Approach more NFV/SDN oriented

The Slice-based Federation Architecture (SFA) 2.0 (2) high level interface specification draft was published in July 2010, before the imminence of current technologies such as Network Function Virtualization and Software Defined Networking. As defined in the draft, a "resource (RSpec) describes a component in terms of the resources it possesses and constraints and dependencies on the allocation of those resources" and the lifecycle is defined elsewhere.

In previous projects, as stated in the above sections, i.e. FED4FIRE[5], the resource definition was done in different steps: the RSpec defined the actual resource to be used and the physical location of it while the "automated" execution of the experiment is defined through an OEDL script, executed via OMF[6].

This example shows that the concept of lifecycle of the experiment has gained a more complex definition. Nowadays, the genre of resources could widely vary from each other. SFA was designed for mainly provisioning compute resources (physical or virtual) to the experimenter. The experimenter was then in charge of handling the *lifecycle* of the actual experiment, via SSH or in some cases, via OEDL script, if the platform was providing OMF support.

SoftFIRE, aware of the lessons learned from the past, must adapt its structure to the new technical requirements coming from the new generation of experimenters, maintaining implemented the FIRE functional requirements that are the concrete definition of a FIRE based project.

4.1 The NFV/SDN enabler modelling language: TOSCA

The recent evolution in the technology state of the art brought the Experimenters to include in their experiments and to require from the used platform some key NFV/SDN functionalities. The

[3] https://trescimo.eu
[4] https://wiki.opnfv.org/display/pharos/Pharos+Home
[5] https://www.fed4fire.eu
[6] http://omf.mytestbed.net/projects/omf/wiki/An_Introduction_to_OMF

SoftFIRE platform intends to meet these requirements, while keeping on provisioning the FIRE required features.

The OASIS industry group defined in November 2013 a standard called Topology and Orchestration Specification for Cloud Applications (TOSCA) that aims to provide an efficient and precise tool able to model the cloud applications and associated IT services having a particular focus on telecom operators requirements. This industry related standard is suitable for modeling NFV and SDN applications (described in addition in the *TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0* specification draft) (3) and it has been chosen by the SoftFIRE consortium to be the reference modelling language for defining Experiments (*Topology Template*) as a set of different kind of resources (*Node Template*) strictly formalized by the SoftFIRE *Node Type* definition[7].

As a matter of fact, TOSCA excellently adapts to experiment definition purposes, maintaining the new NFV/SDN requirements met, keeping abstract the experiment configuration and making the definition highly portable.

4.2 The SoftFIRE Middleware

The decision of adopting a new information model for the northbound API inducted the SoftFIRE Middleware to evolve in order to meet all the SDN/NFV requirements and all the functional requirements that are embedded in a FIRE project. The Middleware was redesigned in an extremely modular architecture, defined in Figure 2, allowing the platform to handle an infinite number of different resource types.

[7] http://docs.softfire.eu/etc/softfire_node_types.yaml

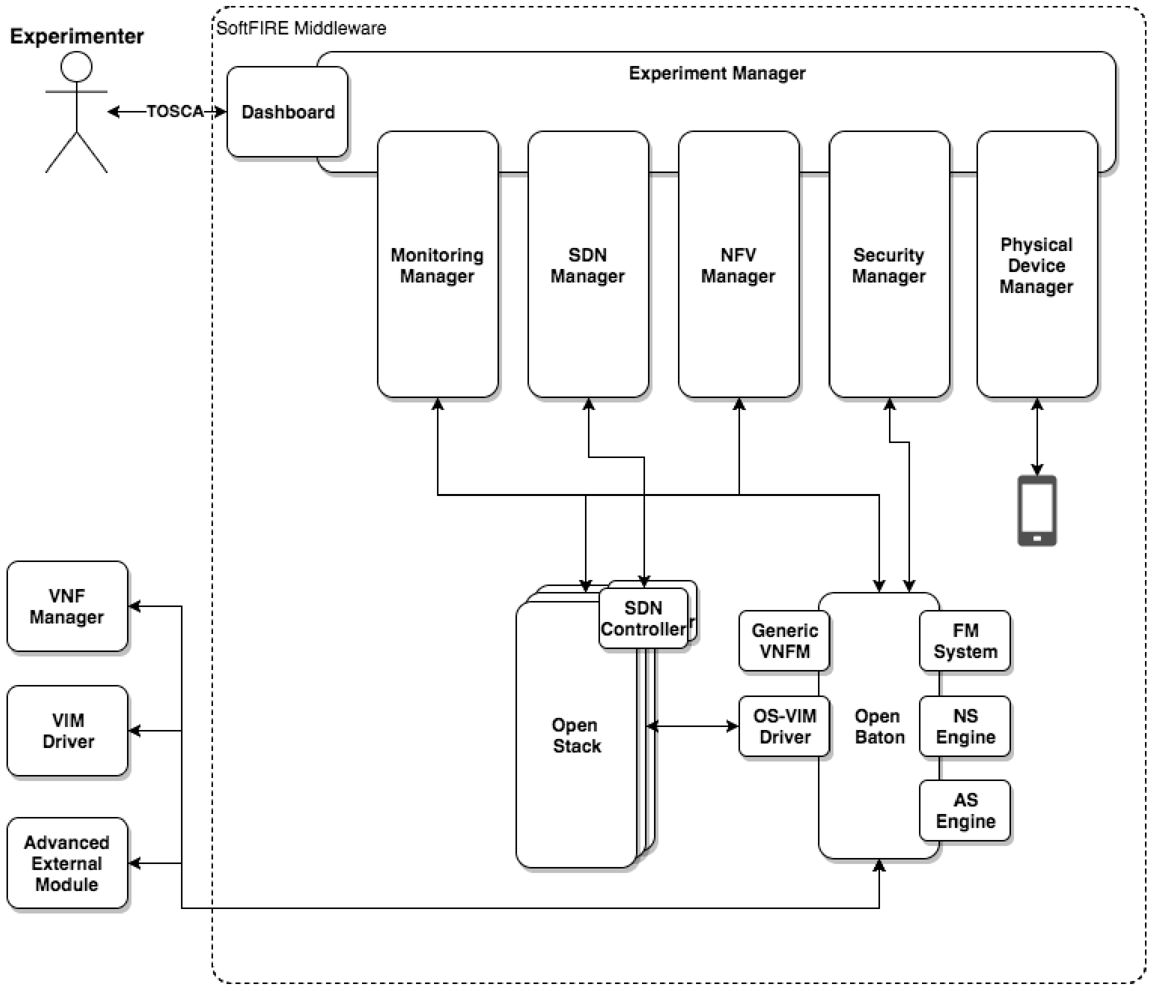

Figure 2: SoftFIRE middleware architecture

Albeit the northbound API is TOSCA standard, the experimenter will interface mainly with a dashboard that has the task of simplification of the interaction with the platform. However, the Experimenter can make use also of some CLI tools that provide more programmability to the platform.

The Experimenter can naturally and easily extend the current platform. This is possible in many ways:

- Extending the *SoftFIRE Middleware* by implementing:
  - A resource specific manager
  - An Open Baton external module
  - An Open Baton VNF Manager
  - An Open Baton Monitoring plugin
- Extending the *SoftFIRE Infrastructure* layer, by integrating:
  - A Openstack API compliant private cloud into the SoftFIRE infrastructure
  - A SDN Controller
  - Any non-Openstack based VIM and its Open Baton VIM Driver

As shown in Figure 2, there are different Managers. The main one is the Experiment Manager (EM) and then there are (at the moment) five sub managers in charge of one specific kind of resource. The resource types that are currently managed inside the SoftFIRE Middleware are therefore five: *NFV resources*, *SDN resources*, *Security resources*, *Physical resources* and *Monitoring resources*.

The Experiment Manager delegates the operations on a specific resource to a specific manager. The foreseen operations mainly reflects the FIRE experimenter operations:

- List resources (*resource discovery*)
- Provide resources
- Terminate resources
- Validate resources
- Refresh resources

Thus, each manager implements these API and knows how to validate, provide and terminate the specific resource. In addition to these functions, we also included a registration, a deregistration and an *update status* method. Each manager must first register to the Experiment Manager and provide the list of resources it is exposing. Some managers also handle some types of resources that are not static and that change status and value after the deployment. For that reason, an *update status* method can optionally be implemented by each manager and that aims to update the status of an already deployed resource.

4.2.1. NFV Manager

The NFV Manager is in charge of managing any TOSCA node of type ***NfvResource*.** A NFV resource represents a Network Service (NS) as defined in the ETSI NFV specification[8]. The TOSCA Node Type definition is as follows:

```
NfvResource:
    derived_from: eu.softfire.BaseResource
    description: "Defines a NFV resoruce request in the SoftFIRE Middleware"
    properties:
      ssh_pub_key:
        required: false
        type: string
      file_name:
        required: false
        type: string
      nsd_name:
        type: string
      testbeds:
        entry_schema:
          description: "mapping between vnf types and testbed. Or 'all':<testbed_name>
for all in one"
          type: string
```

[8] http://www.etsi.org/technologies-clusters/technologies/nfv

```
        type: map
```

Figure 3. Definition of the NfvResource in TOSCA language

Where the fields represent:

- **resource_id**: The resource id identifies a resource available from the resource discovery. However, for this particular type of resource, the resource id can also not be included in the list of available resources in case the experimenter wants to upload his own *NfvResource*
- **testbeds**: a map where you can define the testbed where each VNF will be deployed.
- **nsd_name**: the name of the NS
- **file_name**: in case the preconfigured NS are not sufficient for your experiment you can upload your own NS in CSAR format and place it in the Files folder. This field contains the name of the file

The possible NSs that the platform can actually deploy is infinite, hence the NFV Manager is the only manager able to handle a *resource_id* that is not included in the list of available resources. In this specific case, the experimenter has to define the NS, following the TOSCA NFV (3) profile and the Open Baton specification (4) and include it in the experiment definition.

4.2.2. SDN Manager

The SDN Manager is in charge of managing access to the SDN resources provided by some of the SoftFIRE testbeds. The SDN manager keeps track of the SDN Controller API endpoints and provides managed access to them through the **SDN proxy** services. It is not sustainable for an open system like SoftFIRE to provide direct access to the SDN Controller API to the experimenter. For that reason, the SDN Proxy filters and provides controlled access to the SDN Controller API.

4.2.3. Physical Device Manager

The Physical Device Manager (PDM) focuses its efforts on guaranteeing access to a physical resource. Generally, most of the physical resources available in an infrastructure require the user to physically be in the physical resource location in order to be able to use it. In the case of these types of resources, the PDM only provides the reservation and configuration methods. In the SoftFIRE infrastructure some physical resources, are made remotely available by a simple dashboard, in the case of these resources, the PDM returns to the Experimenter also the access to this dashboard.

4.2.4. Security Manager

The Security Manager inside the SoftFIRE Middleware makes available to the Experimenter a series of security related functionalities that he might decide to include and use within his activities on the SoftFIRE platform. Here is the list of the available features.

- The Experimenter can statically configure the *SecurityResource* by means of its descriptor
    - The Experimenter can enable logs collection from his resource
    - The Experimenter can statically configure some rules on his resource
- The Experimenter can dynamically configure the resource once it has been deployed
- The Experimenter can see the resources logs in a web dashboard
- The Experimenter can perform searches among the resources logs in a web dashboard
- The Experimenter can see statistics related to the resources logs in a web dashboard

A *SecurityResource* is a commonly used security agent that the Experimenter can include in his experiment. He can access and configure it through a static initial configuration, included in the TOSCA description of the Resource, or, once deployed, through a ReST interface that exposes its main services. The Experimenter can also ask the *SecurityResource* to send its log messages to a remote log collector, which makes them available in a simple web page reserved to him. The Experimenter could easily access it through its web browser and check the behaviour of all his security agents, and to check the statistics. The Experimenter can obtain the *SecurityResource* in two different formats:

- As an agent directly installed in the VM that he wants to monitor. The system will provide him a script that the Experimenter has just to run inside the VM. It will be already configured as required in the TOSCA description of the resource. The output of the script will provide to the Experimenter information on how to access the deployed resource (URLs, etc.)
- As a standalone VM. The *SecurityResource* will be deployed directly by the Security Manager in the testbed chosen by the Experimenter. The Security Manager will take care of the initial configuration of the resource. The Experimenter has to set up on his own the redirection of the network traffic that he wants to control in the deployed VM (by means of tunneling or SDN capabilities).

The *SecurityResource* NodeType is described as follows:

```
SecurityResource:
    derived_from: eu.softfire.BaseResource
    description: "Defines a Security agent to be deployed. More details on [docu_url]"
    properties:
        resource_id:
            type: string
            required: true
        testbed:
            type: string
            required: false
        want_agent:
            type: boolean
            required: true
        logging:
            type: boolean
            required: true
        allowed_ips:
            type: list
            entry_schema:
                type: string
            required: false
        denied_ips:
            type: list
            entry_schema:
                type: string
            required: false
        default_rule:
            type: string
            required: true
```

Figure 4. SecurityResource NodeType definition in TOSCA

4.2.5. Monitoring resources

The Monitoring Manager provides monitoring as a resource to the Experimenter. For experimenters requiring monitoring resources, the Monitoring Manager provides a dedicated instance of a state of the art monitoring server. All NFV resources requested by the experimenter will be configured in order to provide monitoring information to the monitoring server. The experimenters receive full administrations rights on the monitoring server, in order to be able to configure it according to the specific needs of the experiment.

4.3 Experiment Definition

As explained above, the experiment is defined using the TOSCA standard. In particular, the Experimenter has to create a CSAR[9] zip file containing all the necessary files and definitions for letting the Experiment Manager (EM) manage the resources included in the experiment.

The SoftFIRE Experiment CSAR (5) is composed by three main folders: Definitions, Files and TOSCA-Metadata.

4.3.1. TOSCA-Metadata

The TOSCA-Metadata folder contains the TOSCA.meta file and the Metadata.yaml file. The TOSCA.meta file must contain the reference to the experiment definition. The Metadata.yaml contains experiment meta information regarding the name of the experiment and the start and end date.

4.3.2. Definitions

The Definitions folder contains the experiment yaml description file that must follow a specific structure. The SoftFIRE experiment yaml file must contain the TOSCA definition version (“*tosca_simple_yaml_1_0*”) and the *imports* section must be specified because the EM will only accept specific node types defined in the node type definition file[10]. Each node type specifies a *resource_id* that must be chosen from the list of available resources (resource discovery). The node name is arbitrary. Each node type can have some additional properties as defined in the previous sections.

[9] http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd01/TOSCA-Simple-Profile-YAML-v1.0-csprd01.html#_Toc430015789

[10] http://docs.softfire.eu/etc/softfire_node_types.yaml

4.3.3. Files

The Files folder contains an inner CSAR defining a NS as specified by the TOSCA for NFV profile[11] (3). This is only used in case the NFV Resource you want to deploy is not one of the available one. This means that the Experimenter can upload his own specific resource and be able to deploy it in a second moment.

5 Conclusion

The goal of this White Paper is to promote within the NFV/SDN/5G community discussion and suggestion on how to fully support the technological capabilities of these technologies by creating a middleware capability that can be used to ease the creation, the execution and the monitoring of relevant experiments. This extension of goals of SoftFIRE is requiring considerable development effort and it is important to have the support of the community also for collecting requirements or suggestion for further improvement of this effort.

It is quite relevant to create a viable and possibly extendible framework that helps in executing experiments on this kind of platforms. The approach and the choice of open interfaces and description languages that fully fit with NFV/SDN are considered as relevant advantage and value by the project. The created infrastructure could be further extended by introducing new capabilities and new resources and related solutions. The new architecture of the Experiment Manager is general enough to include new resources that are not foreseen in the current state of the art (meaning that new FIRE projects can use the same architecture and that same software implemented).

In addition, it allows a lot of flexibility to Experimenters for putting together different functionalities such as monitoring, security and last but not least SDN control capabilities. This could provide to the experiment a rich set of functionalities. However there is not obligation on the experiment side to use all of them at once. The different Managers can be used in a progressive manner starting from the NFV Manager to be used for virtualized software functions and then to extend the functionalities according to needs and ability to program and govern them.

The intended benefits of this effort can be summarized in this way:

- Availability of a flexible and rich middleware specifically designed for managing NFV/SDN technologies based on industry oriented open APIs (TOSCA)
- Integration of security and monitoring capabilities into the middleware framework
- Possibility of integrating also physical resources so that new resources can be added and integrated into the platform. This could be particularly relevant for future activities aiming at extending the platform towards a richer set of 5G network resources
- Support in terms of protocols and APIs the experimenters that choose only specific kind of resources in a particular location for a dedicated amount of time. This gives the flexibility to experiment in the small and then increase the number of functionalities to be linked and used within a specific experiment and development.

[11] http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html

The SoftFIRE project is keen to receive suggestions, inputs or criticisms to the chosen approach. We believe these topics need discussion and awareness within the growing NFV/SDN community.

Bibliography

1. **Trescimo Project.** Deliverable D3.3: TRESCIMO Prototype v2. [Online] 11 2015. https://trescimo.eu/wp-content/uploads/2015/11/D3.3_v1.0.pdf.

2. **Peterson, Larry, et al.** Slice-Based Federation Version 2.0. [Online] 7 2010. http://groups.geni.net/geni/raw-attachment/wiki/SliceFedArch/SFA2.0.pdf.

3. **Lipton, Paul, et al.** TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0. [Online] 5 11, 2017. http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html.

4. **OpenBaton Team.** OpenBaton: TOSCA virtual network function template. [Online] Jun 1, 2017. https://openbaton.github.io/documentation/tosca-vnfd/.

5. **Lipton, Paul, et al.** TOSCA Simple Profile in YAML Version 1.0. [Online] August 27, 2015. http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd01/TOSCA-Simple-Profile-YAML-v1.0-csprd01.html.

List of Acronyms and Abbreviations

Acronym	Meaning
5G	Fifth Generation Mobile Network
API	Application Programming Interface
EM	Experiment Manager
MANO	Management and Orchestration
M2M	Machine-to-Machine
NFV	Network Function Virtualisation
NFVO	Network Function Virtualisation Orchestrator
ODL	OpenDaylight
SDN	Software Defined Network
SEM	SoftFIRE Experiment Manager
TOSCA	Topology and Orchestration Specification for Cloud Applications
VM	Virtual Machine
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VPN	Virtual Private Network