SOFTFIRE

Software Defined Networks and Network Function Virtualisation Testbed within FIRE+

Programmability with NFV and SDN in SoftFIRE

Programmability features offered to NFV/SDN experimenters by the SoftFIRE federated testbed

March 2018

Table of Contents

List of Figures

List of Tables

1 Introduction

Network Functions Virtualisation (NFV) [1][2] and Software Defined Networking (SDN) [3] technologies are changing the game in the telecommunications market using concepts and approaches from the Information Technology (IT) world. The main goal of NFV is full automation of network services, including service lifecycle management, whereas SDN is more focused on programmable networking functions that complement NFV, particularly for the Fifth Generation (5G) mobile networks and applications.

The NFV platform provided by the EU SoftFIRE project [4], i.e. its federated virtualisation testbed, exposes a number of its Application Programming Interface (API) sets to enable programmability of the platform by its experimenters. In this document, the aim is to provide an overview of these sets of API where experimenters can extend the platform, add new features to it, or simply enable their experiments. . The document has a top-down approach, first providing the general overview of the platform in terms of its programmability, and then presents some of the platform components which expose their API.

First, the SoftFIRE Experiment Manager (SEM) [5] provided by SoftFIRE is presented, which is the high level tool that enables experimentation on the platform, and links an experiment with several software components, each of which act as a manager of a special experimentation purpose. Then, particular focus is given to the SDN features of the platform, which make it possible to implement additional networking functionalities.

2 Programmability on the SoftFIRE Middleware: The Experiment Manager

The SoftFIRE Experiment Manager is the first point of interaction between the experimenter and the platform. The main feature provided to the user by this tool is the reservation of the platform's virtualisation resources their experiments, which is done by uploading CSAR (Cloud Service ARchive) [6] packages to the middleware.

2.1 Virtual resources provided to experimenters by the SEM

An experimenter can create virtual network functions (VNF) and network services (NS). Besides these, the SoftFIRE platform also provides many out-of-the-box resources that the experimenter can easily use and instantiate, which can interact and be integrated with their custom VNFs.

The virtualisation resources provided by SoftFIRE are listed in Table 1 below:

Table 1. Virtualisation resources provided by SoftFIRE.

Resource Name	Resource Type	Description
Monitor	Monitoring Resource	This resource permits to deploy a Zabbix [7] server that can be used to monitor all the experiment virtual machines (VM).
Iperf	NFV Resource	iPerf [8] is a tool for active measurement of the achievable data throughput in IP networks. It supports tuning of various parameters related to timing, buffers, and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test, it reports the available bandwidth, packet losses, and some other parameters.
Open5GCore	NFV Resource	Open5GCore [9] is a prototype implementation of pre-standard 5G networks. The software has been available since November 2014. Open5GCore represents the continuation of the OpenEPC project [10] towards R&D testbed deployments. It has been used over the years in multiple projects as a reference vEPC implementation.
Open IMS core	NFV Resource	The Open IMS Core [11] is an open source implementation of IMS Call Session Control Functions (CSCFs) and a lightweight Home Subscriber Server (HSS), which together form the core elements of all IMS and Next Generation Network (NGN) architectures as specified today within 3GPP, 3GPP2, ETSI TISPAN and PacketCable [12]. The four components are all open source software (e.g. the SIP Express Router (SER) or MySQL).
FOKUS-cell	Physical Resource	The Physical LTE Cell at Fraunhofer FOKUS [13] that is connected to the Open5GCore NS. This LTE Cell is a physical resource, which can be reserved using the Physical Device Manager.
5GIC UEs	Physical Resource	The three mobile phones located in indoor cabinets in the 5GIC building in the University of Surrey, with Android OS. The phones can be remotely reserved as a physical resource via the UE Manager, and then remotely controlled. The UE Manager is reachable via the Physical Device Manager, when the user chooses to reserve UEs as a physical resource.
Ericsson Opendaylight SDN Controller	SDN Resource	OpenDaylight (ODL) Controller [15] API endpoint for the Ericsson Testbed. It allows the experimenter to create and edit its traffic flows. ODL uses its OpenFlow [16] Plugin REST API [17] to program hardware and software switches.
FOKUS OpenSDNCore Controller	SDN Resource	OpenSDNCore Controller [18] JSON-RPC API endpoint for the Fraunhofer FOKUS Testbed.

Resource Name	Resource Type	Description
Firewall	Security Resource	This resource creates an instance of UFW [19] firewall. It can be deployed as a standalone VM or as an agent directly installed on a VM of the experiment. It also provides an easy REST API to edit its rules.
Suricata	Security Resource	This resource creates an instance of Suricata [20] Network Intrusion Prevention System (NIPS), which is a free opensource network threat detection engine. It can be deployed as a standalone VM or as an agent directly installed on a VM.

Most of these resources are fully programmable using their API. Below the links to the API documentation of each of these resources is listed in Table 2:

Table 2. APIs for some SoftFIRE software resources.

API Name
Zabbix JSON-RPC API [21]
Firewall UFW REST API [22]
Suricata log and packet acquisition APIs [23][24]
pfSense REST API [25]
Opendaylight OpenFlow plugin REST API [17]
OpenSDNCore JSON-RPC API [18]
OpenStack SFC API [42][43][44]

The experiment CSAR package created by the experimenter contains some YAML [26] files with the description of the experiment, the resources needed, and the reservation scheduling. Users can define their custom NSes and VNFs using the NS and VNF descriptors of Open Baton NFVO, which are fully compatible with the simple profile defined by the TOSCA (Topology and Orchestration for Cloud Applications) language [27], for NFV. How to create this file is well documented in [28].

2.2 Architecture of the SEM

The SEM architecture has a high modular design, with loosely coupled components that can be easily attached or detached from the main SEM. This kind of architecture was well defined in this manner because, the SEM should not be just a piece of software to manage experiments, but it was developed with the idea of creating a framework usable also in the future for similar projects.

Currently, the SEM is composed of the following modules:

- **SDN Manager** manages SDN resources
- **Security Manager** for the Security resources
- **NFV Manager** is in charge of providing NFV functionalities to the middleware
- **Monitoring Manager** provides experimenter monitoring resource access
- **Physical Device Manager** handles the access to the physical resources

The components communicate between them using gRPC protocol [29].

As the architecture is modular, it is possible to add new managers to the SEM; this is presented later in Section 6. The architecture of the experiment manager is shown in Figure 1.

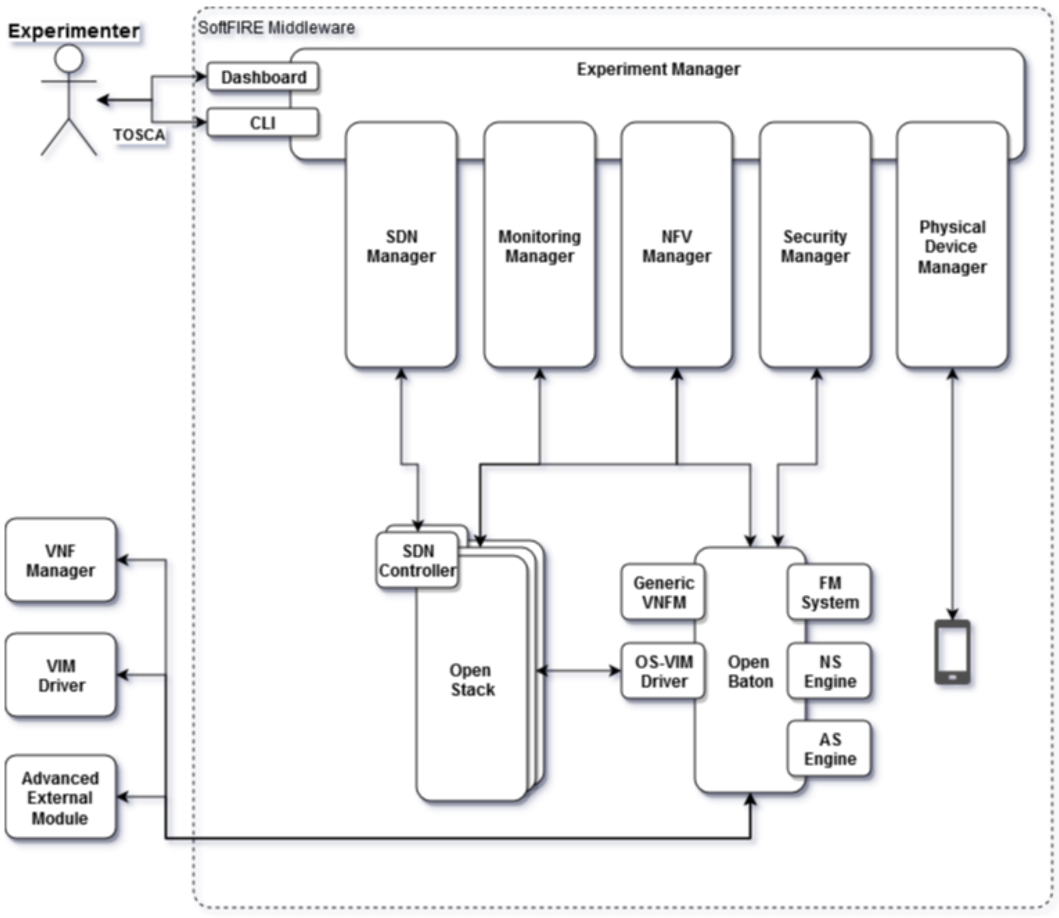

Figure 1: The Experiment Manager Architecture.

As shown in the figure, the NFV Manager is the middleware module that interacts with the NFVO OpenBaton, and the SDN Manager is the module that communicates with multiple SDN controllers, each provided by a different testbed in SoftFIRE. The Security and Monitoring managers, however, are more related with complementary features of the middleware, i.e. providing extra resources to experimenters.

When experimenter includes a security resource in their experiment package, the SEM uses the Security Manager to deploy the specified resource, which is one of the following:

- Ubuntu UFW (Uncomplicated Firewall) [19]
- Suricata Server (Open Source Network Threat Detection Engine) [20]
- pfSense Server (Open Source FreeBSD based FW/Router Server) [30]

Similar to the Security Manager, the Monitoring Manager provides monitoring tools as resources to experiments if requested. If requested by an experimenter, the SEM triggers and action with the Monitoring Manager, which in turn provides a Zabbix server already installed and configured inside the user tenant in order to monitor the experimenter's VMs.

Requesting security resources require the experimenter to redirect their VMs' traffic through the deployed security service, which also means that the experimenter needs to create custom routing tables in each involved VM. This provides flexibility to the experimenter as to how to design the processing of their traffic. The experimenter may also choose to program traffic flows using the SDN controller provided by the testbed(s) where the VMs are deployed. This requires requesting SDN resources as part of the experimentation package. The component testbed ADS [31] also provides service function chaining (SFC) [32][33] modules.

Finally, the Physical Device Manager allows the experimenter to reserve an LTE Femto cell physically located at the FOKUS testbed, or get remote access to the smartphone devices connected to the LTE cell at 5GIC, University of Surrey.

3 Network Service programmability using TOSCA

The SoftFIRE NFV platform is fully based on the ETSI MANO [42] compliant open source orchestrator Open Baton [35]. Open Baton orchestrates Network Services consisting of a set of VNFs. The deployment of the NFV resources can be done through the usage of TOSCA Network Service Descriptors (NSD) for network services [36], and Virtual Network Function Descriptors (VNFD) for VNFs. The descriptors allow to define the topology of all the NFV resources that must to be created in order to instantiate the required Network Service.

The NS must be packaged in a CSAR file, which is a zip archive with extension ".csar" that contains the following directory structure:

```
├── Definitions
|   └── myNSD.yaml
├── Scripts
|   ├── install.sh
|   └── (VNF TYPE)
|         └── script.sh
```

```
└── TOSCA-Metadata
    ├── Metadata.yaml
    └── TOSCA.meta
```

Figure 2. Example CSAR file for an NS.

In the above example, the file myNSD.yaml is to contain the NSD for the requested NS. The CSAR file essentially describes the directory structure and lists the necessary files for the network service. These include the scripts to be run for lifecycle events.

In order to better a TOSCA NS descriptor, it is essential to clarify some key concepts, which are listed in Table 3:

Table 3. Network Service terms and descriptions.

Term	Description
Network Service (NS)	A Network Service is composed of a set of Virtual Network Functions
Virtual Network Function (VNF)	A Virtual Network Function contains one or more Virtual Deployment Units (VDU)
Virtual Deployment Unit (VDU)	A VDU is a profile of a virtual machine. Scale-in and Scale-out actions are performed on VDU instances by the orchestrator.
Virtual Link (VL)	VL is an abstraction of a network on which the VDUs are attached
Connection Point (CP)	CP represents the connection between a Virtual Link and a VDU

In this table, a hierarchical dependency of different NFV concepts are listed top to down. Based on this information, it is possible to have a glance at a sample NSD in YAML format, as seen in the below code snippet:

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0
description: Example of NSD

metadata:
  ID: dummy-NS
  vendor: Fokus
  version: 0.1

topology_template:

  node_templates:

    dummy-server:
        type: openbaton.type.VNF
        properties:
          vendor: Fokus
          version: 0.1
          endpoint: dummy
          type: server
          configurations:
            name: server-configurations
            configurationParameters:
```

```
          - key: value
          - key2: value2
      deploymentFlavour:
        - flavour_key: m1.small
    requirements:
      - vdu: VDU1
    interfaces:
      lifecycle:
        INSTANTIATE:
          - install.sh
          - install-srv.sh

dummy-client:
  type: openbaton.type.VNF
  properties:
    ID: x
    vendor: Fokus
    version: 0.1
    type: client
    deploymentFlavour:
      - flavour_key: m1.small
    endpoint: dummy
  requirements:
     - vdu: VDU2
  interfaces:
      lifecycle: # lifecycle
        INSTANTIATE:
          - install.sh
        CONFIGURE:
          - server_start-clt.sh

VDU1:
  type: tosca.nodes.nfv.VDU
  properties:
    scale_in_out: 1
  artifacts:
    VDU1Image:
      type: tosca.artifacts.Deployment.Image.VM
      file: ubuntu-14.04-server-cloudimg-amd64-disk1

VDU2:
  type: tosca.nodes.nfv.VDU
  properties:
    scale_in_out: 2
  requirements:
    - virtual_link: CP2
  artifacts:
    VDU1Image:
      type: tosca.artifacts.Deployment.Image.VM
      file: ubuntu-14.04-server-cloudimg-amd64-disk1

CP1:
  type: tosca.nodes.nfv.CP
  properties:
    floatingIP: random
  requirements:
          - virtualBinding: VDU1
          - virtualLink: private

      CP2: #endpoints of VNF2
        type: tosca.nodes.nfv.CP
        requirements:
          - virtualBinding: VDU2
          - virtualLink: private

      private:
        type: tosca.nodes.nfv.VL
        properties:
          vendor: Fokus

  relationships_template:
    connection_server_client:
      type: tosca.nodes.relationships.ConnectsTo
      source: dummy-server
      target: dummy-client
      parameters:
          - private
```

Figure 3. Code Snippet 1 - A sample NSD in YAML format.

Using the YAML format, the NSD is essentially human-readable file, and most of its attributes are self-explanatory. For more information about each attribute, there is documentation at the Open Baton website [36].

As presented in Figure 3, the NS descriptor for an NS allows an experimenter to declare a section with all the parameters that need to be provided at NS instance creation time, which will then be referenced and used as environment variables in the lifecycle scripts of the VNF(s) of that NS.

When a VNF needs to know some attributes of another VNF in an NS, the experimenter must declare a 'relationships_template'. To explain this concept, in the following example, two VNFs are considered: a client VNF (VNF-A) and a server VNF (VNF-B), as shown in Figure 4. In this example, one of VNF-A's parameter , i.e. its IP address, is required by VNF-B, so that it can be used when VNF-B is started. This is a typical example of dependency between two VNFs, in this case VNF-B depends on VNF-A's parameter, and VNF-A is the "source" of the parameter which is needed and hence is "targeted at" VNF-B.

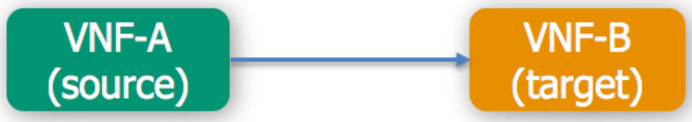

Figure 4. VNF Dependencies

To support ease of programming, Open Baton already provides three out-of-the-box dependency parameters, as listed in Table 4.

Table 4. OpenBaton's out-of-the-box dependency parameters.

Out-of-the-box-parameter	Usage of variables in scripts	Value
The IP address	`$< network_name >`	The IP address assigned to the virtual machine belonging to the network with name
The floating IP address	`$< network_name >_floatingIp`	The floating IP address assigned to the virtual machine belonging to the network with name
The hostname	`$hostname`	The hostname assigned to the virtual machine

3.1 VNF lifecycle management

A powerful programmability feature provided by the Open Baton orchestrator is the VNF Lifecycle management, which allows to declare and run scripts to manage and control lifecycle events VNFs.

According to the ETSI MANO specification [34][42], a VNF has the following lifecycle events:

Table 5. VNF lifecycle events, as defined by ETSI.

Lifecycle event	Description
INSTANTIATE	The instantiation of the corresponding VNF.
CONFIGURE	This lifecycle event happens after VNF instantiation, and is necessary for specific configuration tasks that can only be performed after the VNF starts. For instance, if the VNF depends on other VNFs, and needs to fetch their parameters that are only available after the VNF has been instantiated (e.g. IP addresses).
START	This event occurs after the instantiation and configuration.
STOP	This event occurs during the stopping of the corresponding VNF.
TERMINATE	This event occurs during the termination of the corresponding VNF.
SCALE_IN	When the VNF is the target of a VNF component (VNFC) on which a scale-in operation is performed.
SCALE_OUT	When the VNF is the target of a VNFC on which a scale-out operation is performed.

The use of the correct scripts in every lifecycle event of a VNF makes it possible to fully automate the creation of the VNF, and the graceful termination of the VNF. In particular, the SCALE_IN is a key status that must be considered in cases where the VNF is a cluster composed of many VNF components (VNFC).In this lifecycle event, before terminating one of the VNFCs, it is possible to notify all the VNFCs that are part of the cluster, notifying that the particular VNFC will no longer be available. For example, if the VNF is a Cassandra cluster [37], before terminating one node of the VNFCs, it is necessary to gracefully remove that particular

VNFC (a node in the cluster) from the cluster in order to be able to move all its data to the other VNFCs (the remaining set of nodes in the cluster).

3.2 Using software configuration and management tools

The correct method to setup an experimenter VNF is to create all associated VMs from a generic image, like the Ubuntu 16.04, and then make sure that each instance installs and then configures whatever software is required. This can be achieved with bash scripting, yet SoftFIRE strongly recommends to use configuration management tools, such as puppet or ansible, which are more user-friendly than bash scripting. The NFVO used in SoftFIRE is fully compatible with these tools; all that is needed to put Puppet Manifest or Ansible Playbook files inside the “scripts” folder (please see Code Snippet 1 in Figure 3) and then apply them with just a one-line script.

In the following, an example is provided for the use of Ansible.

3.2.1. Setting up a web server VNF using Ansible

The following example demonstrates how to setup a VNF called ‘WebServerVNF’, and then configure it as a web server using an Ansible playbook in yaml format.

```
    WebServerVNF:
      type: openbaton.type.VNF
      properties:
        ID: x
        vendor: Fokus
        version: 0.1
        type: WebServerType
        deploymentFlavour:
          - flavour_key: m1.medium
        endpoint: WebServer
      requirements:
         - vdu: WebServerVDU
      interfaces:
          lifecycle: # lifecycle
            INSTANTIATE:
              - apply-webserver-ansible-playbook.sh
```

Figure 5. Code Snippet 2 - WebServer VNF.

The Ansible playbook yaml file (see Figure 7) as well as the one-line script (see Figure 6) that runs it will then be placed inside the CSAR package, in the folder called `scripts/WebServerType`. The following are these two files to be included in the experiment CSAR package:

- `apply-webserver-ansible-playbook.sh`
- `webserver-playbook.yaml`

```
#!/bin/bash
ansible-playbook -i "localhost," -c local webserver-playbook.yml
```

Figure 6. Code Snippet 3 - The one-line script that runs the Ansible playbook for the web server.

```
---
- hosts: local
  tasks:
   #Install Apache2 and notify the handler StartApache2
   - name: Install Nginx
     apt: name=apache2 update_cache=yes state=latest
     notify:
      - StartApache2

   #Clone git repo to apache2 www folder
   - name: Git clone
     git: >
       repo: 'https://mygit.example.org/path/to/repo.git'
       dest: /var/www

  handlers:
   #Ensure the service Nginx is running and enabled
   - name: StartApache2
     service: name=apache2 state=started enable=true
```

Figure 7. Code Snippet 4 - Ansible playbook yaml file contents, to create a web server VNF.

4 NFV Orchestrator Extensibility

Open Baton NFV orchestrator has a modular architecture that can be easily extended for supporting various use cases. All its components communicate through a RabbitMQ Messaging service. The Open Baton architecture is shown in Figure 8.

SoftFIRE experimenters have the opportunity to use all the features provided by the NFVO and its modules, such as the Autoscaling Engine or the Fault Management System. Some experimenters in the Waves of Experiments [38] on the SoftFIRE platform have also extended these modules, e.g. in the case of the Autoscaling Engine, and experiment deployed a new module based on machine learning algorithms.

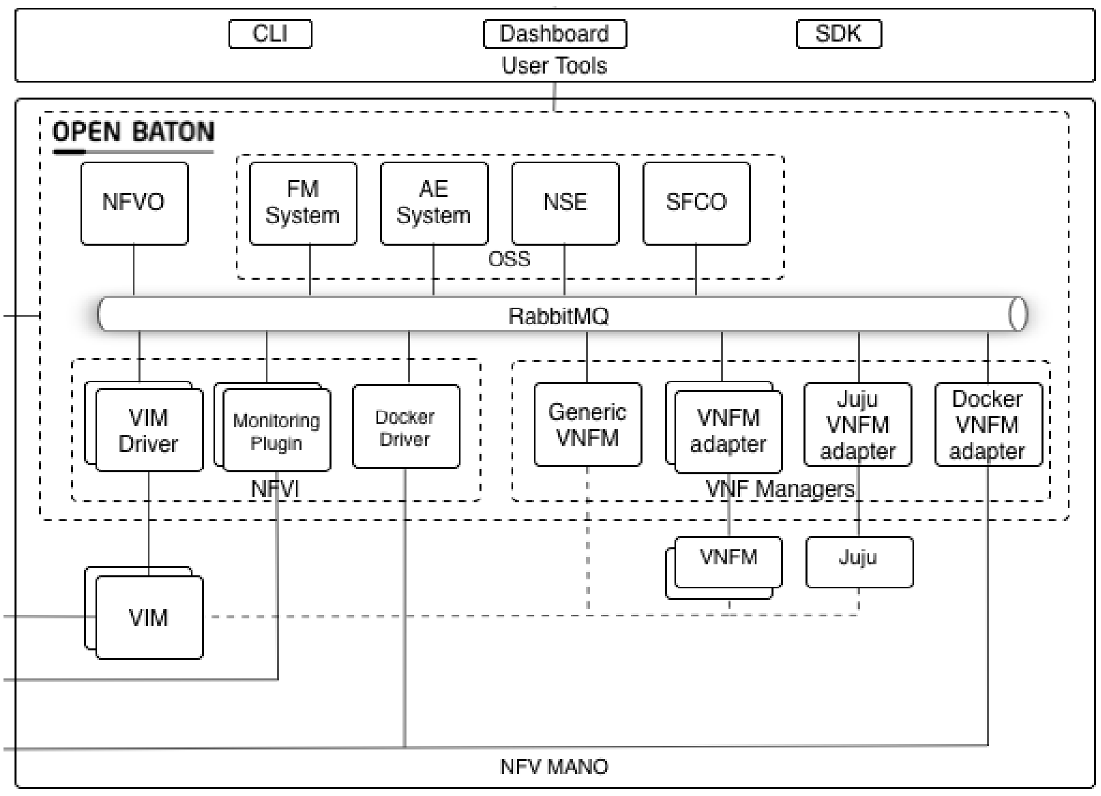

Figure 8. Open Baton Architecture.

Extension of the orchestrator can be accomplished using the Open Baton SDKs to develop new modules or to extend the already existing ones. Below is the list of all Open Baton SDKs:

Table 6. Open Baton SDKs for extending the orchestrator with new modules or extending the already existing modules.

SDK name	Description	gradle artifact
NORTHBOUND SDK	It is used by all the OpenBaton modules like the Fault Management (FM) System and the Auto-scaling (AS) Engine. It allows to create a client that can receive events from the Open Baton RabbitMQ Messaging Service and execute actions triggered by the events.	org.openbaton:sdk
VIM DRIVER SDK	It is used to implement new drivers to instantiate VNFs on a new VIM. It has been used to implement the Openstack [39] VIM driver or Amazon VIM driver. It provides an abstract class with all the methods that must be implemented.	org.openbaton:plugin-sdk
VNFM SDK	It can be used to develop a new VNF Manager to suite the specific needs of a VNF.	org.openbaton:vnfm-sdk

5 SDN Controller Access

The Ericsson and FOKUS component testbeds in SoftFIRE each have an SDN controller available to SoftFIRE experimenters, to provide a more fine-grained programmability of the open virtual switches [40] that connect the VMs. The Ericsson testbed uses the OpenDaylight SDN controller [15] which is one of the most popular open source SDN controllers. The FOKUS testbed leverages on OpenSDNCore [18] as its SDN controller.

An experimenter accessing the SDN controller can implement advanced networking features. For instance, the experimenter can implement a mirroring feature to dump the incoming/outgoing packets of a VNF, or dynamic packet diversion based on a set of flow-match criteria defined according to the OpenFlow protocol. Both SDN and OpenSDNCore controllers can be programmed with HTTP requests, for the Ericsson testbed the experimenter can access the REST API of Opendaylight's OpenFlow plugin [17]. OpenSDNCore provides a JSON-RPC interface.

As documented in SoftFIRE online documentation [41] after an experimenter requests access to the SDN controller, three flow tables are assigned to the experimenter, and all the traffic of the experimenter's VMs pass through the first assigned table. This is illustrated in Figure 9. In this way, the experimenter can edit/add all the flows regarding only the VMs belonging to the experimenter. This mechanism ensures tenant isolation in the SDN space. In SoftFIRE, the flow redirection mechanism was implemented in the SDN Manager, and is coupled with an ODL Proxy and an Open SDN Core proxy, which are in charge of filtering HTTP requests sent to the ODL and Open SDN Core, respectively. Such filtration is necessary to ensure that an experimenter's HTTP request is only for the tables assigned to that experimenter, and to avoid any potential malicious activity.

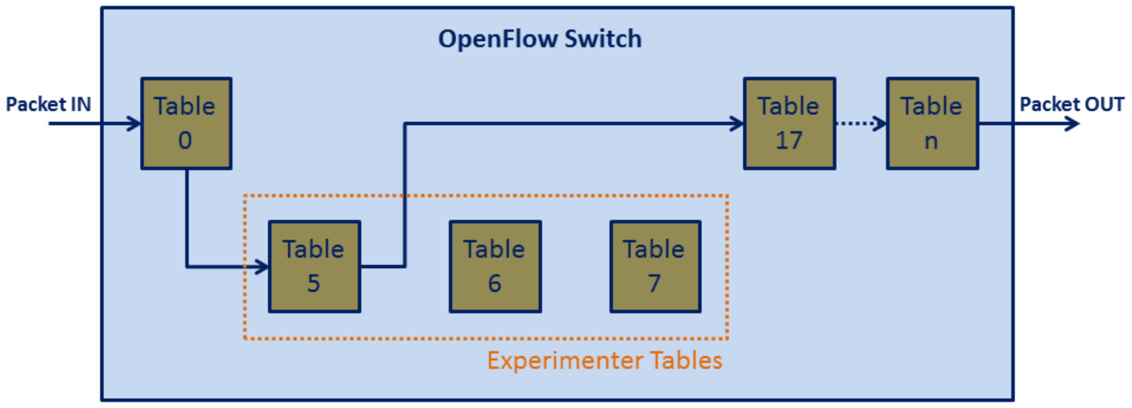

Figure 9. OpenFlow tables assigned to an experimenter.

5.1 Openstack SFC

Besides the SDN controllers described in Section 5, another feature offered by SoftFIRE to control an experimenter's traffic flows is provided by the ADS component testbed, which are the Openstack Service Function Chain (SFC) modules [42]. Service function chaining extension for OpenStack Networking can be found in [43], and contributor API are in [44]. This solution allows experimenters to define service function chaining with a higher abstraction layer than the SDN controllers. Using flow classifiers with some flow matching rules based on packet attributes, experimenters can change packet flow descriptions, for example when suspicious traffic from an IP address is detected, it can be dynamically redirected to another VNF that can check the suspected traffic using Deep Packet Inspection (DPI) tools.

6 Extending the SoftFIRE Middleware

SoftFIRE provides all of its middleware source code on publicly available github repositories [45] in order to allow future developments and reuse of the code on similar experimentation platforms. All the middleware managers were built from python softfire-sdk, which is publicly available on github as source code [46] and also on PyPi Python Packages Repository [47] as binary.

Figure 10 shows the interactions between the SoftFIRE components and each Resource Manager.

The platform can be extended by adding custom manager software.

In order to develop your own manager software, you need to install the sdk with the following command:

```
$ pip install softfire-sdk
```

The new manager needs to extend the abstract class AbstractManager with methods that implement the message calls shown in Figure 10.

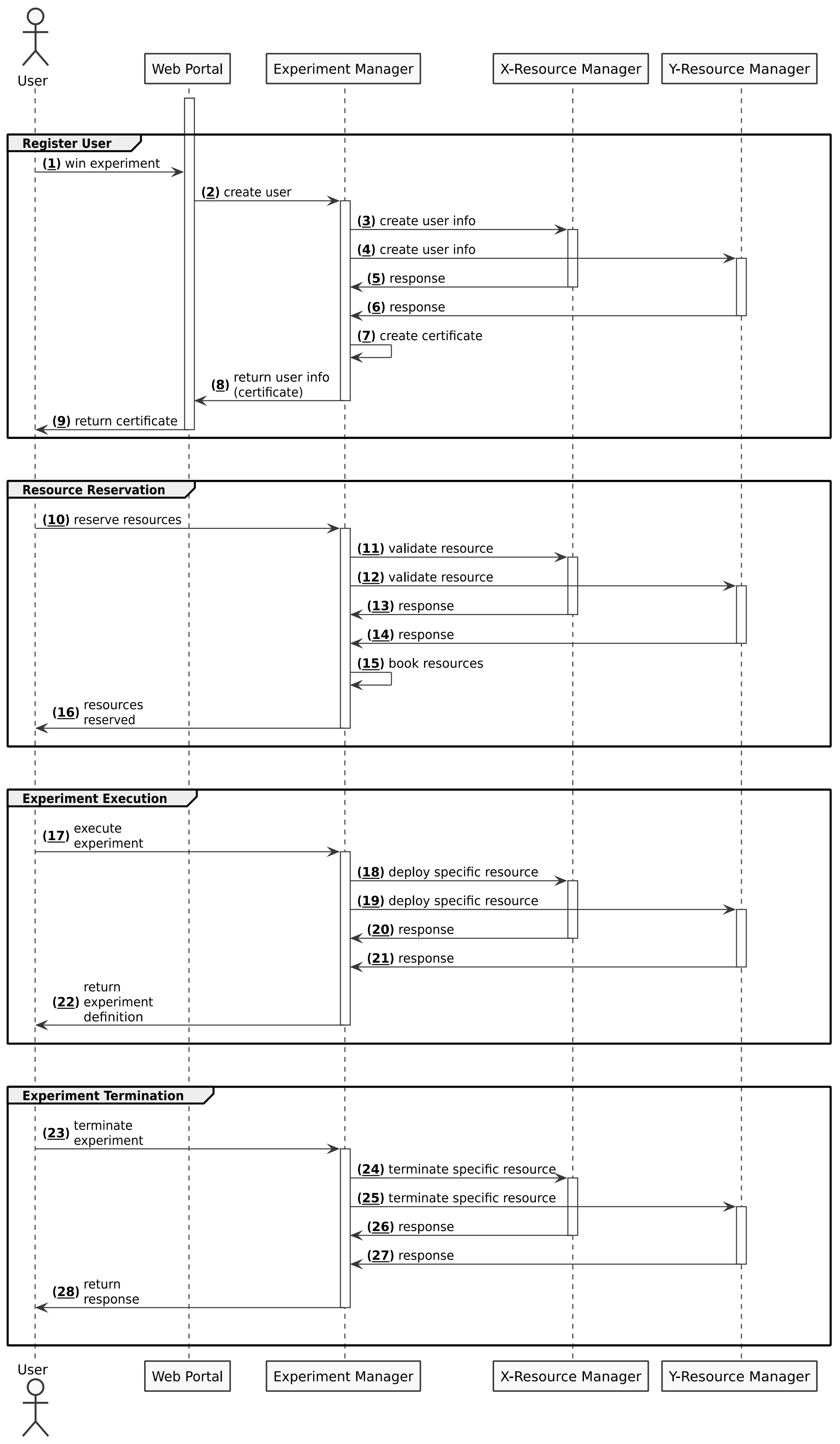

Figure 10. SoftFIRE Resource Manager Generic Sequence Diagram (between two Resource Managers: X and Y)

7 Conclusions

Extensibility and programmability are the two key concepts which form the foundations of the approach in building the SoftFIRE platform. Programmability is pervasive in the project from many points of view. An experimenter can automate their deployments using the provided API and templates, or can extend the platform with newly implemented modules which are not provided by the middleware; for instance, a machine-learning based auto scaling engine has been built by an experimenter, which works with the middleware software. The platform uses many programming languages like Java, Python, Javascript, various integrations tools like Jenkins, and the open source infrastructure manager Openstack that has become the de-facto standard for NFV platforms. However, NFV technologies are still evolving, and are influenced by new technologies that emerge from the datacentre and IT markets. Such influence from datacentre markets has been at a more rapid pace than what has been observed in the telecommunications market. This has been particularly noticed during the lifetime of the project, when new tools have been tested by experimenters. The modular structure of the SoftFIRE middleware has enabled experimenters to implement their requested functionalities, and is envisioned to be a generic approach for future NFV federation projects. The fundamental aspects of the project has been the use of free open source software, and their integration, whilst also enabling programmability of the platform to users.

Bibliography

[1] “Network Function Virtualisation: State-of-the-Art and Research Challenges”, Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, Raouf Boutaba, *IEEE Communications Surveys & Tutorials*, vol 18, no 1, 236-262, September 2015.

[2] "Network Functions Virtualisation— Introductory White Paper", ETSI, 22 October 2012, retrieved 20 June 2013.

[3] “Software-Defined Networking: A Comprehensive Survey”, Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, *Proceedings of the IEEE*, vol 103, no 1, pp 14-76, January 2015.

[4] EU SoftFIRE project, https://www.softfire.eu/

[5] SoftFIRE Middleware, http://docs.softfire.eu/softfire-middleware/

[6] The TOSCA Cloud Service Archive (CSAR), https://www.oasis-open.org/committees/download.php/46057/CSAR%20V0-1.docx

[7] Zabbix, http://www.zabbix.com/

[8] Iperf, https://iperf.fr/

[9] Open5GCore, https://www.open5gcore.org/

[10] OpenEPC project, https://www.openepc.com/

[11] Open IMS Core, http://www.openimscore.com/

[12] PacketCable, https://www.cablelabs.com/security-docs/packetcable/

[13] Fraunhofer Fokus, FUSECO Playground, https://www.fokus.fraunhofer.de/go/en/fokus_testbeds/fuseco_playground

[14] 5G Innovation Centre, University of Surrey, http://www.surrey.ac.uk/5gic

[15] OpenDaylight, https://www.opendaylight.org/

[16] OpenFlow, https://www.opennetworking.org/sdn-resources/openflow

[17] OpenDaylight OpenFlow Plugin, https://wiki.opendaylight.org/view/OpenDaylight_OpenFlow_Plugin:User_Guide

[18] Open SDN Core, https://www.opensdncore.org/

[19] Uncomplicated Firewall, https://help.ubuntu.com/community/UFW

[20] Suricata, https://suricata-ids.org

[21] Zabbix API, https://www.zabbix.com/documentation/3.4/manual/api

[22] UFW API, https://github.com/softfire-eu/softfire-eu.github.io/blob/sitecode/docs/etc/firewall_api.adoc

[23] Suricata log API, https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Log_API

[24] Suricata packet acquisition API, https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Packet_Acquisition_API

[25] Pfsense API, https://github.com/ndejong/pfsense_fauxapi

[26] YAML, http://yaml.org/

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

[28]Experiment definition in SoftFIRE, http://docs.softfire.eu/experiment-definition/
[29]gRPC, Google, [Online]. Available at: https://grpc.io/
[30]pfSense, [Online]. Available at: https://www.pfsense.org/
[31]Assembly Data System (ADS) NFV lab testbed, https://www.assembly.it/
[32]Service function chaining architecture, IETF, [Online] Available at: https://tools.ietf.org/html/rfc7665
[33]Service function chaining, OPNFV, [Online], Available at: https://wiki.opnfv.org/display/sfc/Service+Function+Chaining+Home
[34]ETSI MANO specification, http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
[35]OpenBaton, https://openbaton.github.io/
[36]Network Service Descriptors in OpenBaton, https://openbaton.github.io/documentation/ns-descriptor/s
[37]Apache Cassandra, http://cassandra.apache.org/
[38]SoftFIRE project experiment waves, https://www.softfire.eu/open-calls/
[39]OpenStack open source cloud computing software, https://www.openstack.org/
[40]Open Virtual Switch, http://openvswitch.org/
[41]SoftFIRE online documentation, docs.softfire.eu
[42]OpenStack service function chaining modules, https://docs.openstack.org/ocata/networking-guide/config-sfc.html
[43]https://docs.openstack.org/networking-sfc/latest/
[44]https://docs.openstack.org/networking-sfc/latest/contributor/api.html
[45]SoftFIRE Middleware Github Repositories, https://github.com/softfire-eu
[46]SoftFIRE SDK Github Repository, https://github.com/softfire-eu/softfire-sdk
[47]SoftFIRE SDK PyPi Package Repository, https://pypi.python.org

List of Acronyms and Abbreviations

Acronym	Meaning
3GPP	Third Generation Partnership Project
5G	Fifth Generation Mobile Network
5GIC	5G Innovation Centre
ADS	Assembly Data System
API	Application Programming Interface
AS	Auto-Scaling
CSAR	Cloud Service ARchive
CP	Connection Point
CPN	Control Plane Node
CPU	Central Processing Unit
DPI	Deep Packet Inspection
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
EU	European Union
FM	Fault Management
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IT	Information Technology
LTE	Long Term Evolution
LTE-A	Long Term Evolution Advanced
MANO	Management and Orchestration
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	Network Function Virtualisation Orchestrator
NGN	Next Generation Networks
NIPS	Network Intrusion Prevention System
NS	Network Service

NSD	Network Service Descriptor
ODL	OpenDaylight
OS	Operating System
OVS	Open Virtual Switch
PoP	Point of Presence
RAN	Radio Access Network
REST	Representational State Transfer
RPC	Remote Procedure Call
SCTP	Stream Control Transmission Protocol
SDK	Software Development Kit
SDN	Software Defined Network
SEM	SoftFIRE Experiment Manager
SFC	Service Function Chaining
SFCO	Service Function Chaining Orchestrator
SIP	Session Initiation Protocol
TCP	Transport Control Protocol
TISPAN	Telecoms & Internet converged Services & Protocols for Advanced Networks
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDP	User Datagram Protocol
UE	User Equipment
UFW	Uncomplicated Firewall
URL	Uniform Resource Locator
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VM	Virtual Machine
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFM	Virtual Network Function Manager
YAML	YAML Ain’t Markup Language

SoftFIRE

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