The Sixteenth Advanced International Conference on Telecommunications
AICT 2020

Charms and Virtual Network Functions Primitives Experiments using Open Source MANO Framework

Andra Ciobanu, Cosmin Conțu, Eugen Borcoci
University POLITEHNICA of Bucharest
andraciobanu90@yahoo.com
cosmin.contu@elcom.pub.ro
eugen.borcoci@elcom.pub.ro

AICT 2020, September 27, 2020 to October 01, 2020 - Lisbon, Portugal

Contents

1. Introduction
2. RELATED WORK-OSM ARCHITECTURE AND FUNCTIONS
3. VNF CONFIGURATIONS AND CONCEPTS, VNF PRIMITIVES AND CHARMS
4. PROXY CHARM BUILD EXAMPLE IN OSM AND BUG FIX EXAMPLE
5. Conclusions

1. Introduction

- **Network Function Virtualization (NFV)** is a strong technology to support the development of flexible and customizable virtual networks in multi-tenant and multi-domain environment. Open issues still exist for architectural, interoperability, design and implementation aspects.
- The **contributions of this paper** consist in introducing Open Source MANO OSM framework, present deeper knowledge about Virtual Network Function (VNF) charms and primitives.
- The **objective** of this deeper knowledge is accomplished with a proxy charm experiment. Also, a bug fix for proxy charm is presented and other capabilities and possible limitations in different contexts are going to be examined and sorted out for the future works.
- Towards this aim, **this paper continues the authors work** and develops new functionalities along with the previous ones and integrates them in a complex network topology using OSM. This work can help the developers implementing NFV systems based on OSM

2. RELATED WORK-OSM ARCHITECTURE AND FUNCTIONS

- **The goal of ETSI OSM** (Open Source MANO) is the development of a community-driven production-quality E2E Network Service Orchestrator (E2E NSO) for telco services, capable of modelling and automating real telco-grade services, with all the intrinsic complexity of production environments
- OSM's approach is:

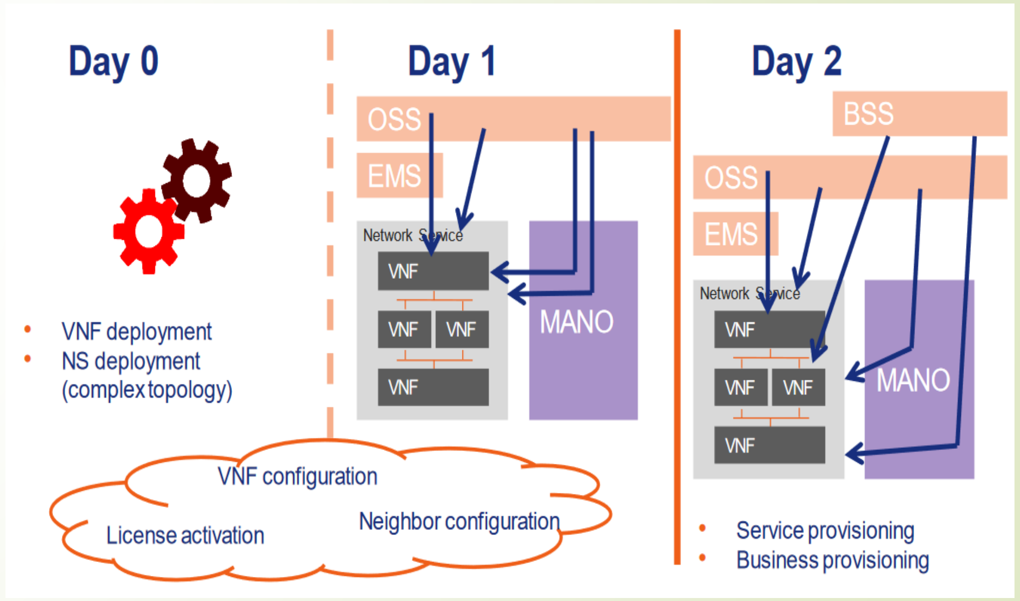
-A **well-known Information Model (IM)**, aligned with ETSI NFV, that is capable of modelling and automating the full lifecycle of network services (NS)

- A **unified northbound interface (NBI)** which enables the full operation of system and the Network Services and Network Slices under its control.

3. VNF CONFIGURATIONS AND CONCEPTS, VNF PRIMITIVES AND CHARMS

- In order to fulfill the complete onboarding process, a VNF Package will be produced and it will be part of the OSM catalogue for its inclusion in a Network Service. The onboarded VNF should aim to fulfil the lifecycle stages.

- To accomplish the lifecycle stages, the resulting package includes all the requirements, instructions and elements which are: basic instantiation (a.k.a. "Day0"), service initialization (a.k.a. "Day1") and runtime operations (a.k.a. "Day2") – see Figure
- In **Day0 stage**, the VNF is instantiated and the management access is established so that the VNF can be configured at a later stage
- The main objective of the **Day1 stage** is to configure the VNF so it starts providing the expected service
- The main objective of **Day2** is to be able to re-configure the VNF so its behavior can be modified during runtime



3. VNF CONFIGURATIONS AND CONCEPTS, VNF PRIMITIVES AND CHARMS

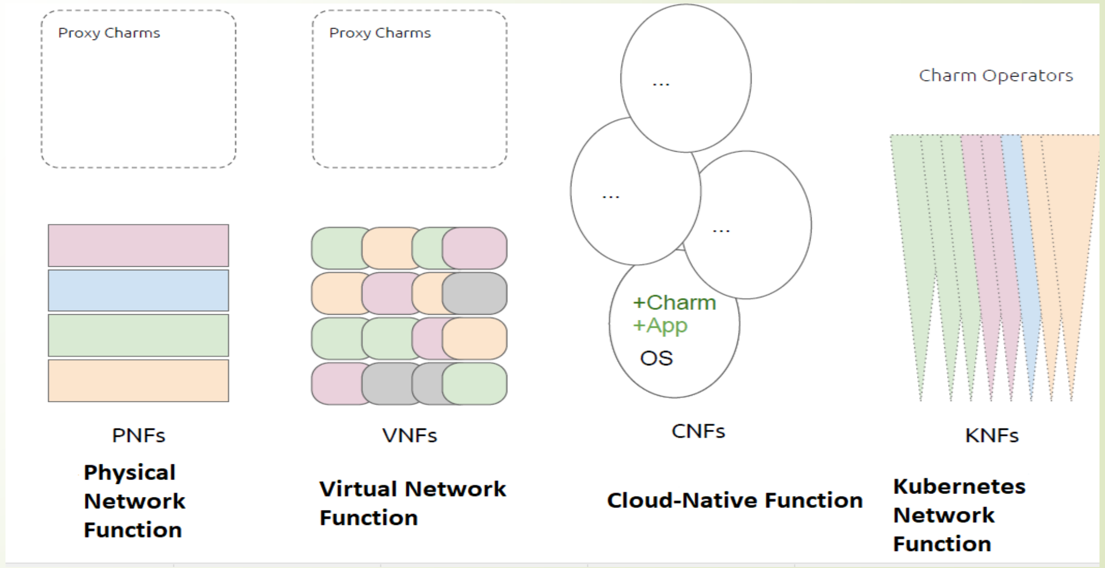
- **Juju** is an open source modeling tool, composed of a controller, models, and charms, for operating software in the cloud
- **A charm** is a collection of actions and hooks that encapsulate the operational logic of an application
- Types of charms:

  -**Proxy**- focus of this paper (used for physical and virtualized network functions; handles day 1 and day 2 configuration; communicates with VNF vis SSH

  - **Machine/Native:** used for Cloud-native Network Functions
- **VNF primitives in OSM**

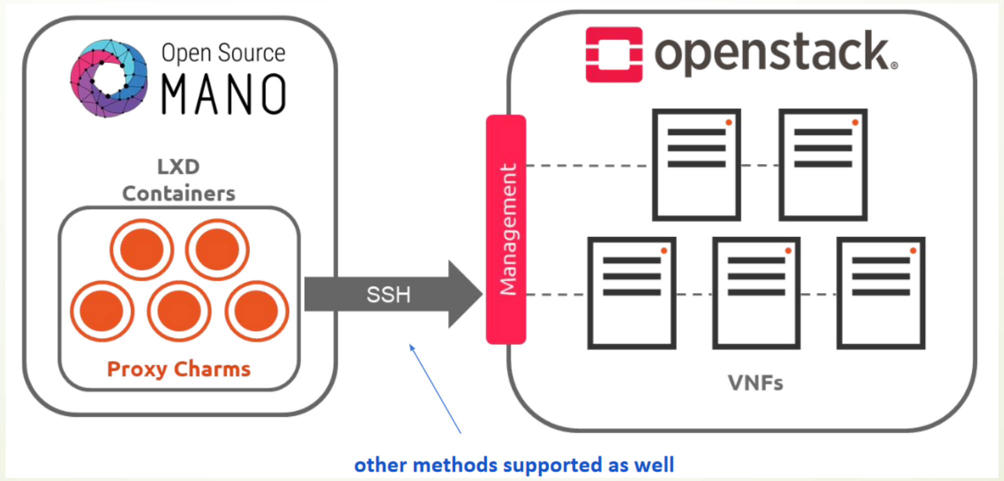
  - Primitives are declared in the VNF Descriptor

  - There is an "initial-config-primitive" and a "config-primitive"



4. PROXY CHARM BUILD EXAMPLE IN OSM AND BUG FIX EXAMPLE

- As opposed to classical “Native charms”, **Proxy charms** run from outside the application. In particular, it runs within a model instantiated in a LXC container, that configures the VNFs through their management interface.
- Proxy charms cover day-1 and day-2 configuration



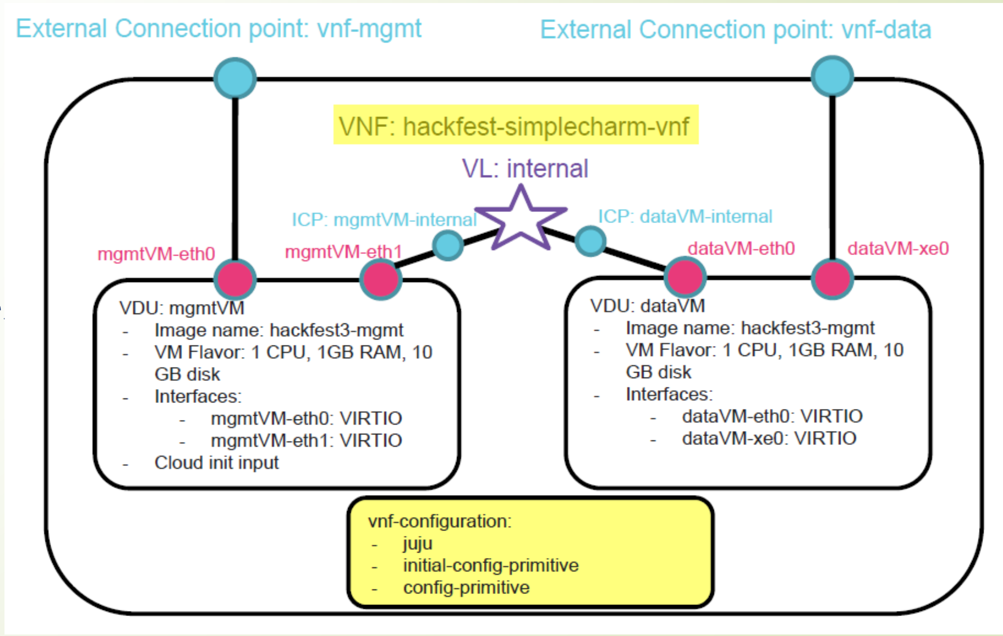
4. PROXY CHARM BUILD EXAMPLE IN OSM AND BUG FIX EXAMPLE

Steps needed to build a proxy charm:

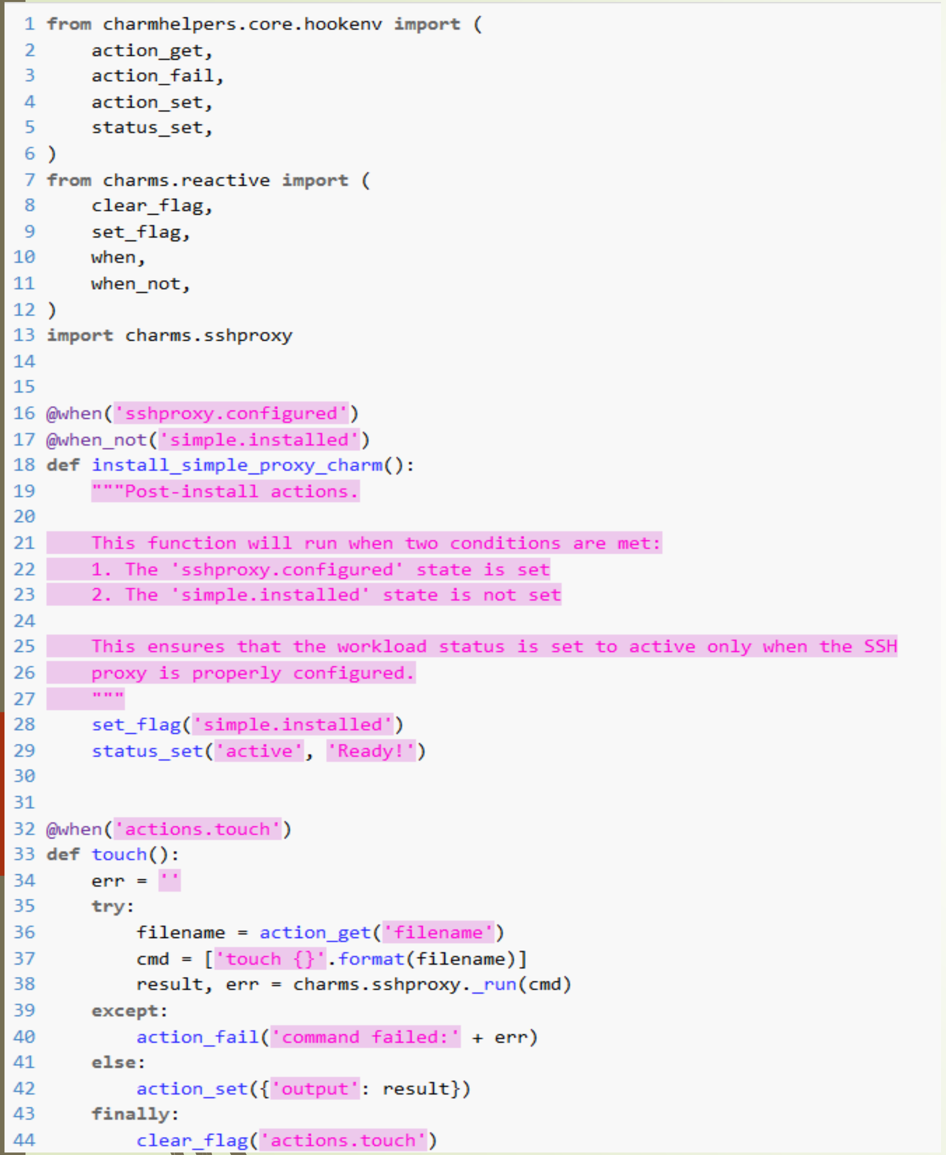
- 1. Setting up a charming environment (*sudo snap install charm --classic #* already installed in using shared OSM)
- 2. Create a Proxy Charm Layer
- 3. Implementing the action (*Append* the implementation of the action to *reactive/simple.py*)

Steps needed to build a VNF package with proxy charm

- 1. Prepare the descriptor with details about VNF requirments
- 2. Prepare cloud-init scripts
- 3. Identify parameters that may have to be provided at later stage to adapt to different infrastructures.



4. PROXY CHARM BUILD EXAMPLE IN OSM AND BUG FIX EXAMPLE



```python
from charmhelpers.core.hookenv import (
    action_get,
    action_fail,
    action_set,
    status_set,
)
from charms.reactive import (
    clear_flag,
    set_flag,
    when,
    when_not,
)
import charms.sshproxy


@when('sshproxy.configured')
@when_not('simple.installed')
def install_simple_proxy_charm():
    """Post-install actions.

    This function will run when two conditions are met:
    1. The 'sshproxy.configured' state is set
    2. The 'simple.installed' state is not set

    This ensures that the workload status is set to active only when the SSH
    proxy is properly configured.
    """
    set_flag('simple.installed')
    status_set('active', 'Ready!')


@when('actions.touch')
def touch():
    err = ''
    try:
        filename = action_get('filename')
        cmd = ['touch {}'.format(filename)]
        result, err = charms.sshproxy._run(cmd)
    except:
        action_fail('command failed:' + err)
    else:
        action_set({'output': result})
    finally:
        clear_flag('actions.touch')
```

- During **charm implementation some problems or bugs** may occur. A common one can appear when the proxy charm presented above has been added was o day-1 stage at the ssh access.
- **The contribution of the paper has been to develop a python script in order to ensure that the workload status is set to active only when SSH proxy is properly configured**.
- The main principal is that charms would set their own workload status
- The implemented script scope is to set the proxy charm's state to active so the LCM knows it's ready to work only when SSH proxy is ok configured
- As it can be seen in figure , the first part invokes the "reactive/simply.py" code, then it sets active status when ssh is configured, then, the last step is to map the action to the commands to be run
- This script imports actions from charmHelpers, reactive and ssh.proxy,then creates a function that mentains the workload status of the proxy charm when SSH is configured

5.Conclusions and future work

- This paper continued the work from our previous one and got deeper into the structure of a VNF and network service from descriptors creation until management setup and instantiation
- The chosen framework is OSM
- The scope has been to get deeper into charms and work with them
- Find , understand common bugs and issues and work for solutions
- As **future work**, new experiments will be done in OSM and contact with OSM community will be maintain in order to adjust different problems
- Also for **future work**, with help of OSM and charms, scientific contributions for how to test and monitor Quality of Service of network services or service chains in network slicing is followed.

Thank you!
Questions?