X-MANO: An Open-Source Platform for Cross–domain Management and Orchestration

Antonio Francescon*, Giovanni Baggio*, Riccardo Fedrizzi*, Enrico Orsini†, Roberto Riggio*
*FBK CREATE-NET, Trento, Italy; Email: {rriggio,rfedrizzi,afrancescon,g.baggio}@fbk.eu
†Intecs, Milan, Italy; Email: enrico.orsini@intecs.it

***Abstract*—In the recent years several proprietary as well as open–source frameworks for network service orchestration have emerged. Similarly a significant body of literature has been produced investigating both the theoretical and practical aspects of network service orchestration. Nevertheless, currently available frameworks for network service orchestration are designed under the assumption that they can have full control over the resources being orchestrated. Such assumption does not hold any more when there is the need to pool both physical and virtual resources across different technological *and* administrative domains in order to deploy a service. In such a scenario, the different infrastructure providers may be reluctant to provide full access to their resources to a third party. In this paper we present X–MANO, a cross–domain network service orchestration framework. X–MANO is effectively deployment–agnostic and can be used in hierarchical, peer–to–peer and cascading (or recursive) configuration. We validate X–MANO through a proof–of–concept implementation over a multi–domain testbed. Finally, we release all the code under a permissive APACHE 2.0 license making it available to researchers and practitioners.**

***Index Terms*—Network Management, Network Function Virtualisation, Multi-domain orchestration, Multi–technology orchestration, Proof–of–concept.**

I. Introduction

Software Defined Networking (SDN) and network Function Virtualization (NFV) are two of the most promising concepts that are set to bring innovation in the ossified networking landscape. As a matter of fact, both technologies are already heavily used by telecommunications operators to deliver services at a fraction of the cost it would take to run them using dedicated appliances. Nevertheless, future network services [1], [2] already call for a pooling of resources across multiple Infrastructure Providers (InPs).

Multi–domain network service orchestration requires to compose resources (both physical and virtual) across different InPs. However, in order to preserve confidentiality, the different InPs must be allowed to orchestrate their part of the network service according to their own internal administrative policies without having to disclose confidential information, such as traffic matrices and internal topology, to the other InPs involved in the service [3], [4], [5]. As a result, existing NFV Management and Orchestration (MANO) frameworks [6], [7] that assume global network knowledge are not applicable to the multi–domain network service orchestration scenario.

In this paper we introduce X–MANO, the first open–source multi–domain NFV network service orchestrator. X–MANO allows to deploy network services across different administrative as well as technological domains. X–MANO consists of a confidentially–preserving interface for inter–domain federation and a set of primitives enabling cross–domain network service life–cycle programmability. Said primitives tackle all the aspects of *cross–domain* network service provisioning including on–boarding, scaling, and termination. We validate the proposed federation interface and programming abstractions by implementing them in a Proof–Of–Concept (PoC) prototype and by using it to deploy a video transcoding service in a multi–domain environment. Finally, we release the X–MANO implementation under a permissive APACHE 2.0 license making it available to researchers and practitioners[1].

The rest of the paper is organized as follows. In Sec. II we present the related work. The X–MANO architecture and interfaces are described in Sec. III. Our reference implementation of the X–MANO interfaces and its functional validation are presented in, respectively, Sec. IV and in Sec. V. Finally, we draw our conclusions in Sec. VI.

II. Related Work

The functionalities for the virtual resource management and network service orchestration have been standardized by ETSI in [8]. Several software implementations have been developed following the ETSI reference framework and are available under an open–source license.

In the T-NOVA project [9], an NFV orchestrator platform, called TeNOR, has been developed. The purpose of TeNOR [10] is to provide and orchestration platform for the automated provision, configuration, monitoring and optimisation of network services over virtualised infrastructures.

Another software tool for NFV orchestration is Open Baton [6] which is an ETSI NFV compliant Network Function Virtualisation Orchestrator (NFVO). Open Baton has the objective of providing a compliant implementation of the ETSI NFV specification. Open Baton is easily extensible,

978-1-5090-6008-5/17/$31.00

[1]On line resources available at: https://github.com/5g-empower/x-mano

integrates with OpenStack, and provides a plug-in mechanism for supporting additional virtual infrastructure managers.

Open Platform for NFV (OPNFV) [7] is a recently started project initiated by the Linux Foundation. Its goal is speeding up the NFV adoption through the implementation of an integrated and open reference platform for NFV orchestration. It aims at creating an open ecosystem for network service developers. OPNFV has an important participation of industrial players and is based on the ETSI NFV reference model.

Finally, Open Source Mano (OSM) [11] is another open source project that provides a practical implementation of the ETSI reference NFV architecture. OpenMano has been released by Telefonica Labs under an APACHE 2.0 license.

The orchestration of network services across multiple domains raises new challenges which in time impose technological constraints, and domain–specific constraints, such as confidentiality of the available resources. On the other hand the frameworks mentioned above have been designed for the orchestration of a single domain in which full access of the resources is granted to the NFVO and are thus not suitable for a multi–domain network service orchestration use case. Indeed, ETSI recently released a report on architectural options taking into account multiple administrative domains [12].

Blueplanet, a division of Ciena, developed a framework for multi–domain service orchestration [13]. Building upon open APIs and model–driven templates, the Blueplanet solution integrates with third–party SDN controllers, element/network management systems, and orchestration platforms. However, being a commercial product, this solution hardly meets the needs of the research community.

To the best of our knowledge, X–MANO is the first initiative to design and implement a complete, open, framework for multi–domain network service orchestration.

III. X–MANO Components and Interfaces

A. Overview

In this section we will introduce the main X–MANO components and interfaces. It is worth noticing that, the X–MANO framework defines only the interfaces for cross–domain orchestration as a result the components described in this section are to be intended as purely logical leaving space for them to be implemented in different ways. As shown in Fig. 1, the X–MANO architecture consists of three logical entities: the Federation Manager (FM), the Federation Agent (FA), and Domain Orchestrator/Manager (DOM).

Figure 1 also illustrates several cross–domain network service orchestration architectures that are supported by the X–MANO framework. This includes hierarchical, cascading, and peer–to–peer architectures. In the latter case (peer–to–peer) the same Federation Manager behaves as master and slave at the same time, depending on request's origin point. In Sec. IV we will describe a proof–of–concept implementation of said interfaces in a realworld prototype.

B. Federation Manager (FM)

The FM is in charge of exposing the resources and the network services available in the federated domains. Resource availability is advertised by each FA to the FM using a VNF Manifest, i.e. a file providing the description, the identifiers, and the monitoring capabilities of the VNFs available in each federated domain. Users compose their multi–domain network services exploiting the resources advertised in the VNF Manifests. The FM is then in charge of splitting the multi–domain network service into many single–domain network services and to push them toward the DOMs trough the appropriate FA. Notice how, in this way, each domain is aware only of a portion of the entire network service.

The FM exposes two interfaces: the Federation Interface (F–If) and the Virtual Domain Interface (VD–If). The former enables communication toward the FA while the latter allows the FM to act like a DOM hiding the complexity of all the underlying federated domains and allowing to expose them as a single domain. The first immediate consequence offered by this approach is the possibility to recursively nest an FM under the control of another FM effectively enabling the creation of Federation of Federators. This approach allows to control the way VNFs and network services are advertised. For example, based on commercial agreements, only a selection of the VNFs available in a given domain can be exposed to another FM potentially with restrictions over their configuration parameters.

C. Federation Agent (FA)

The FA is in charge of retrieving all the information related to VNFs and network services available within one domain and of exposing them to the FM. It is also responsible for translating the requests coming from a FM into DOM–compliant requests and returning to the FM the responses generated by the DOMs. Each FA supports a northbound and a southbound interface. The former is the F–If, while the latter can be either the DOM northbound interface (D–If), whose implementation is domain–specific, or a VD–If.

D. Domain Orchestrator/Manager (DOM)

An entity in charge of all management activities in a given domain. Even if the X–MANO design has been inspired by the ETSI MANO architecture, no constraints are imposed on the D–If except that it must support basic VNF and network service life–cycle management operations (creation, chaining, and deletion). The DOM must also support monitoring capabilities over the instantiated VNFs.

E. Customer Portal

The Customer Portal, is the web frontend of the FM allowing users to interact in an easier way with the federation system and presenting to the users all the needed information in a complete and usable way. The Customer Portal can be split in two different submodules: the REST server and the

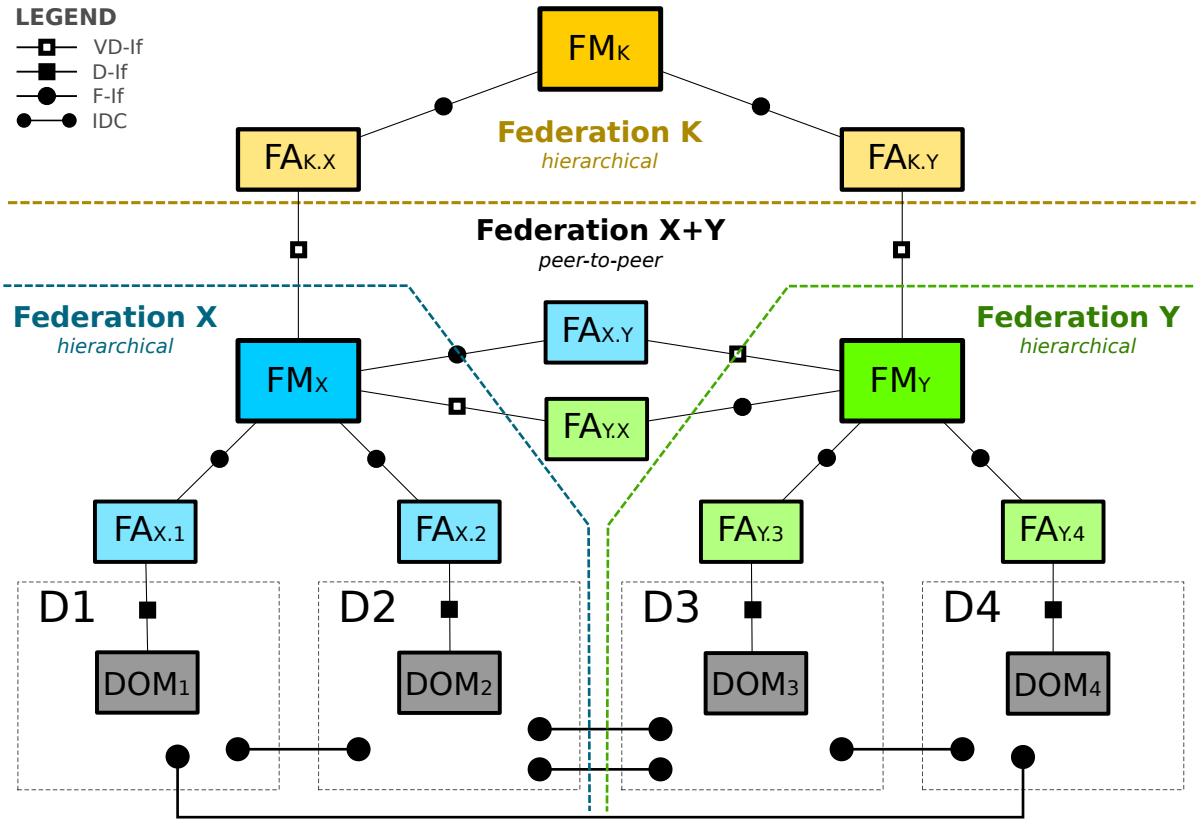


Fig. 1. Example of the different architectures supported by the X–MANO framework. Domains D1, D2, D3, and D4 have some Inter-Domain Connections (IDC) established between them. Each domain has its own DOM managing its local resources. D1 and D2 are federated by FM_X via Federation Agents $\text{FA}_{X.1}$ and $\text{FA}_{X.2}$, respectively. Similarly, D3 and D4 are federated by FM_Y via $\text{FA}_{Y.3}$ and $\text{FA}_{Y.4}$, respectively. FM_X and FM_Y federate each other via $\text{FA}_{X.Y}$ and $\text{FA}_{Y.X}$, forming a peer–to–peer federation, whilst FM_K federates them both in a hierarchical fashion via $\text{FA}_{K.X}$ and $\text{FA}_{K.Y}$, respectively.

front–end graphical user interface (GUI). The first one uses the FM northbound API to operate and retrieve information. The latter is a GUI accessible through any modern web browser. The interface between the GUI and the FM is implemented as a REST interface. The benefits of this approach are that the FM is not GUI dependent, in the sense that any client that can consume a REST service can interact with the FM.

The GUI is implemented in Java using the Google GWT framework which translates Java code into optimized JavaScript code. The GUI is composed of five private areas, or panels, and one login page. Not all the five panels are accessible to all the users. For example the settings panel is accessible only to administrators allowing them to perform management operations like creation and deactivation of users.

The login panel is the only panel accessible by a non–logged user and allows users to authenticate with the system. After login, a user directly accesses the overview panel which is used to provide a view of the user's deployed resources. The catalogue panel contains the VNF catalogues providing the user with a view on the deployable resources. The network service management panel contains all the running network service instances belonging to a particular user. The panel also allows users to launch and manage network service instances. Finally, a statistics panel is available in order to visualize and possibly export the collected monitoring information.

IV. Implementation Details

To demonstrate the usefulness of the X–MANO framework in real–world settings, we developed a proof–of–concept implementation of both a FM and a FA and we used them to deploy and configure a multi–domain video transcoding network service. In this section, we will provide some implementation details about our prototype while in the next section we will report on its performance evaluation.

A. Federation Manager

The FM is developed in Python and consists of different components. The User Account Manager is in charge of authorization, accounting, and authorization. The Network Service Manager is in charge of running the state machine for the network service life–cycle management. The FM uses MongoDB for storing information related to users, multi–domain network service descriptors, and VNF Manifests. The multi–domain network service descriptors are stored in YAML format while the VNF Manifests are stored in JSON format.

B. Federation Agent

The FA is a domain–specific component. In our prototype the FA is implemented in Python and interacts with the Open Baton REST interface to deploy and configure network services. Moreover, the FA stores the VNF Manifests to be advertised to the FM, and retrieves the monitoring information exposed by the domain trough Zabbix [16].

C. X–MANO Interfaces

The interface F–If between FM and FA is implemented using the RabbitMQ scalable message queuing system. RabbitMQ has been chosen because it allows asynchronous bidirectional communication among parties, and because it provides a configurable message broker. Moreover, RabbitMQ exposes a powerful API and has a detailed documentation. Conversely, the VD–If interface has been implemented using the Tornado Web Framework [14]. The main reason for choosing Tornado is its non–blocking network I/O which allows to continue serving incoming requests while the others are being processed. Notice how the D–If is dependent on each DOM implementation which is, in our case, Open Baton. For this reason, the D–If is the REST interface exposed by Open Baton. However, when supported by the DOM, asynchronous interfaces should be preferred.

D. Interconnection of domains

VNF chaining is usually performed using domain–specific techniques. The most common solution for VNF chaining is layer–2 encapsulation, also known as VLAN tagging. However, this approach is not suitable for chaining VNFs across the public Internet since VLAN tagging only works within the same broadcast domain. Another solutions is based on IP tunnelling using for example PPTP, L2TP, L2TP/IPsec, SSTP, or OpenVPN. However, not all the above solutions are suitable for all the federated domains due to security, performance, and compatibility considerations.

Consequently, it appears evident that no standard solutions can be forecast for the realisation of the interconnection of two domains. In order to circumvent this problem we allow each pair of domains to autonomously select an interconnection technology. Once a tunnel has been established, each domain advertises the tunnel endpoints as two VNFs that can be used in order to perform VNF chaining.

V. Evaluation

The goal of this section is to provide evidence on the operations of our X–MANO implementation. The aim is to show that the X–MANO is able to deploy network services with a negligible overhead and without imposing limitations on the underlying domains. In this section we first describe the use case under consideration, then we discuss the evaluation methodology and the performance metrics. Finally we report on the outcomes of the measurements campaign.

A. Video Transcoding Network Service

The network service used in this paper implements a video streaming/transcoding application and is composed by one streaming VNF and one or two transcoding VNFs running on different domains. Both the streaming and the transcoding VNFs have been implemented using the VideoLAN application [15]. Figure 2 depicts the multi–domain network service deployment life–cycle. The deployment consists of three stages. During the INIT stage the resources on the two domains are allocated. In the INIT1 stage the information related to the multi–domain VNFs chaining are collected by the FM. Finally, during the INIT2 stages the video streamer and video transcoder VNFs are started.

The FM enters in the first stage automatically when the network service is launched, while before entering the other stages additional conditions must be verified. Listing 1 reports the section of the multi–domain network service descriptor where the conditions for entering the INIT2 stage are defined. Notice how the FM checks if the VNF chaining has been completed in both domains before transitioning to the stage INIT1 (conditions *domain1_chain_completed* and *domain2_chain_completed*).

By looking at the timeline in Fig. 2, it is easy to notice that the first two stages (INIT and INIT1) involve operations that are run in parallel. While the last stage (INIT2) involves operations that must be executed sequentially. This is due to the fact that the video transcoder VNF has to wait for the video streamer to have started before it can start. This behaviour is confirmed in the multi–domain network service descriptor in the Listing 1, where the *start_streaming* and *start_transcoding* actions are placed in consecutive steps.

It is worth to notice how the FM solved an important issue related to the multi–domain network service orchestration namely the lack of information of the video transcoder about the IP address of the video streamer. The solution is in the multi–domain network service descriptor and in particular in *get source ready* step which retrieves the IP address of the video streamer and stores it in a shared variable. This value is then used as an argument of the *start the transcoder* step. Since the IP retrieval and the video stream initialisation operations do not depend one on another, they are placed in the same step, allowing their parallel execution.

B. Evaluation Methodology

Our testbed consists of two separate OpenStack deployments. For each domain, Open Baton has been selected as DOM. A FA is deployed within each domain and Zabbix is used for retrieving real–time measurements which are collected by the FA and reported to the FM.

For evaluating the proposed work, we first defined different use cases that allowed us to perform a comparison among different federation conditions (including the single–domain case). In particular, we setup the same network service in

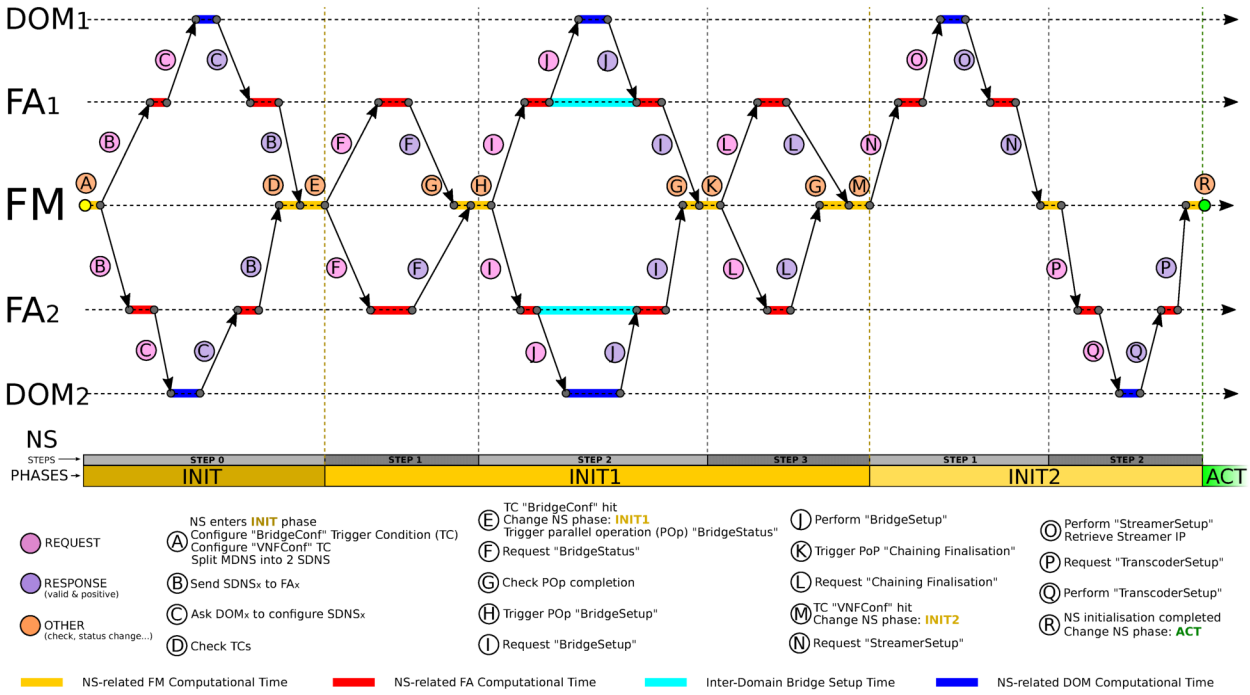


Fig. 2. Example of multi-domain network service setup. Two domains D_1 and D_2, which are managed by DOM_1 and DOM_2 respectively, are federated by FM via FA_1 and FA_2. The network service is composed by a video streaming VNF and by a transcoding VNF. the former is going to be deployed in D_1 while the latter will be deployed in D_2. Please note that the streaming VNF must be started before the transcoding VNF.

three different scenarios. In the first one (SD) we consider a single domain where the service is fully orchestrated by Open Baton. In the second scenario (SD-X) we consider a single domain orchestrated by X–MANO trough Open Baton through. Finally, in the third scenario (MD) we consider the multi–domain case with two Open Baton instances managed through X–MANO.

For each of the three scenarios two network service configurations have been tested one with a single video transcoding VNF and one with two video transcoding VNFs. The measurements that have been collected, are related to the full network service setup time with 2 and 3 VNFs. For the multi–domain case, the VNFs chaining time over the inter–domain link has been also measured. For each of the six test cases the measurements have been repeated 20 times.

C. Results

The results of the measurements are summarized in Fig. 3(a). As it can be seen, the measurements that have been collected show high variance. This behaviour is due to the non–deterministic nature of the underlying computing and networking infrastructure, that supported the experimentations. Nevertheless, the distribution of the network service setup time in the multi–domain case in the case of 3 VNFs is comparable to the single–domain case. Conversely, when only 2 VNFs are involved the median deployment time across two domains seems to be slightly lower than the median deployment time over a single domain. The reason for this behaviour can be ascribed to the fact that in the multi–domain setting less work is required from the DOMs while at the same time the overhead introduced by the X–MANO is negligible.

Fig. 3(b) shows the time required to perform the inter–domain chaining with two and three VNFs. As it can be seen the time required to perform this operation is in the order of a couple of seconds and can be then considered negligible when compared to the full network service setup time.

Finally, we would like to stress that the goal of these measurements was not to report an accurate performance comparison between single–domain and multi–domain network service setup time but rather to be able to claim that: (i) X–MANO does not introduce functional limitations in the single domain network service definition, i.e. it can be transparent with respect to the existing solutions, (ii) X–MANO can effectively deploy network services that span multiple domains, and (iii) X–MANO does not introduce noticeable overhead.

VI. Conclusions

In this paper we introduced a framework, named X–MANO, which allows to coordinate the operations of different single domain orchestrators under a single federation umbrella while at the same time allowing domain administrators to carefully tune the amount of information to be disclosed to such federation layer. Finally, we release the proof–of–concept

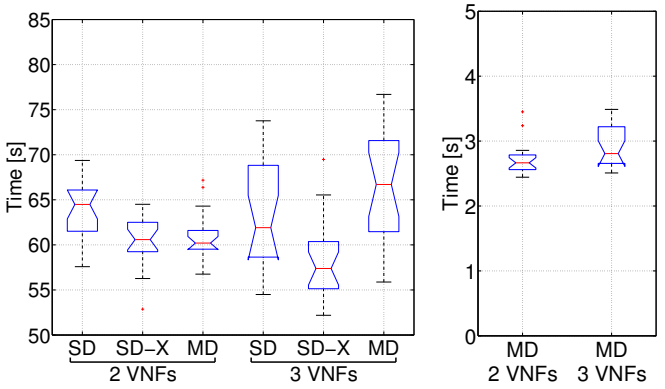
```
Triggers:
 # trigger and steps for stage "INIT2"
 - name: on chaining finalized
   condition:
    and:
     - name: domain1 chained
       metric1:
        type: datastore_type.bool
        value: [domain1_chain_completed]
       metric2:
        type: datastore_type.bool
        value: True
       operator: '=='
     - name: domain2 chained
       metric1:
        type: datastore_type.bool
        value: [domain2_chain_completed]
       metric2:
        type: datastore_type.bool
        value: True
        operator: '=='
- name: domain1 chained
   steps:
    - name: get source ready
      type: steps.vnfs_actions_step
      vnfs_actions:
       - name: video_source
         actions:
          - name: get_ip
            params:
             - name: ifname
               value: dom1_gre
            return_value: ip
          - name: start_streaming
            params:
             - name: videofile
               value: BigBuckBunny_320x180.mp4
             # ... other params
    - name: start the transcoder
      type: steps.vnfs_actions_step
      vnfs_actions:
       - name: video_transcoder
         actions:
          - name: start_transcoding
            params:
             - name: source_ip
               value: [ip]
             # ... other params
    - name: turn mdnsr active
      elaborations:
       - type: steps.elaboration.FMfunction
         name: CHANGE_STATUS
         params:
          - name: mdns_status
            value: active
```

Listing 1. Portion of the multidomain video transcodign NSD.

X–MANO implementation under a permissive APACHE 2.0 license making it available to researchers and practitioners.

ACKNOWLEDGEMENT

Research leading to the results presented in this paper has received funding from the European Union's H2020 Research and Innovation Programme under the Grant Agreement H2020-ICT-644843 (VITAL).



(a) Entire setup. (b) Inter-domain chaining.

Fig. 3. Network service setup time.

REFERENCES

[1] R. Ferrús, H. Koumaras, O. Sallent, G. Agapiou, T. Rasheed, M.-A. Kourtis, C. Boustie, P. Glard, and T. Ahmed, "SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges," *Physical Communication*, vol. 18, Part 2, pp. 95 – 112, 2016.

[2] T. Taleb, A. Ksentini, and R. Jantti, "Anything as a Service for 5G Mobile Systems," *IEEE Network*, vol. 30, no. 6, pp. 84–91, November 2016.

[3] C. Bernardos, L. Contreras, and I. Vaishnavi, "Multi-domain network virtualization," Working Draft, Internet-Draft draft-bernardos-nfvrg-multidomain-01, October 2016.

[4] T. Mano, T. Inoue, D. Ikarashi, K. Hamada, K. Mizutani, and O. Akashi, "Efficient virtual network optimization across multiple domains without revealing private information," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 477–488, Sept 2016.

[5] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *Proc. of ACM VISA*, New Delhi, India, 2010.

[6] "Open Baton." [Online]. Available: https://openbaton.github.io/

[7] "OPNFV: Open Platform for Network Function Virtualization." [Online]. Available: https://www.opnfv.org/

[8] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Management and Orchestration*, Std. ETSI GS NFV-MAN 001, December 2014.

[9] FP7 T-Nova Project. [Online]. Available: http://www.t-nova.eu/

[10] "TeNOR NFV Orchestrator." [Online]. Available: https://github.com/T-NOVA/TeNOR

[11] "Open Source Mano." [Online]. Available: https://osm.etsi.org/

[12] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*, Std. ETSI GS NFV-IFA 009, July 2016.

[13] "Blueplanet, Multi-domain service orchestration." [Online]. Available: http://www.blueplanet.com/products/multi-domain-service-orchestration.html

[14] "Tornado Web Server." [Online]. Available: http://www.tornadoweb.org/

[15] "VLC Media Player." [Online]. Available: http://www.videolan.org/vlc/index.html

[16] "The Enterprise-class Monitoring Solution for Everyone." [Online]. Available: http://www.zabbix.com/