X–MANO: Cross–domain Management and Orchestration of Network Services

Antonio Francescon*, Giovanni Baggio*, Riccardo Fedrizzi*,
Ramon Ferrús†, Imen Grida Ben Yahia‡, Roberto Riggio*
*FBK CREATE-NET, Trento, Italy; Email: {rriggio,rfedrizzi,afrancescon,g.baggio}@fbk.eu
†Universitat Politécnica de Catalunya, Barcelona, Spain; Email: ferrus@tsc.upc.edu
‡Orange Labs, Paris, France; E–Mail: imen.gridabenyahia@orange.com

***Abstract*—The orchestration of network services is a well investigated problem. Standards and recommendation have been produced by ETSI and IETF while a significant body of scientific literature can be found exploring both the theoretical and practical aspects of the problem. Likewise several open–source as well as proprietary tools for network service orchestration are already available. Nevertheless, in most of these cases network services can only be provisioned across a single administrative domain effectively preventing end–to–end network service delivery across multiple Infrastructure Providers (InP). In this paper we present X–MANO, a cross–domain network service orchestration framework consisting in an inter–domain confidentially–presenting federation interface and in an information model for multi–domain network service life–cycle programmability. X–MANO is effectively deployment–agnostic and can be used in hierarchical, peer–to–peer and cascading (or recursive) configuration. We validate X–MANO trough 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 Virtualization, Multi-domain orchestration, Multi–technology orchestration, Proof–of–concept.**

I. Introduction

Driven by the ever increasing demand for new services and applications, telecommunication networks are witnessing a major revolution in both their architecture and service delivery model. In particular, the consideration that not all services are born equal has led to the creation of the term demand–attentive networks [1] highlighting the paradigm shift from a supply–push to a demand–pull service delivery model.

Network Function Virtualization (NFV) is the key technological enabler for demand–attentive networks. NFV is already 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 already call for a pooling of resources across multiple InPs. Examples include mobile operators leveraging on satellite network operators for back–hauling connectivity [2] or content providers aggregating cloud, network, and mobile resources to deliver an end–to-end user experience [3], [4].

One of the biggest challenge in multi–domain network service orchestration is to compose resources from different InPs under a single umbrella framework without imposing requirements or restrictions on the different InPs. In particular each InP shall be allowed to orchestrate its part of the network service according to its 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. As a result, existing NFV Management and Orchestration frameworks [5], [6] that assume global network knowledge are not applicable.

In a multi–InPs scenario a network service may span across different administrative as well as technological domains. Notice how the different domains can leverage on specific orchestration frameworks addressing the management requirements of that particular technology. Moreover, even if technological heterogeneities can be abstracted away under a single network orchestrator, network services deployed across different *administrative* domains will face unique challenges due to the lack of network service orchestration frameworks capable of enabling inter–InP communications.

In this work we take a step in the direction of enabling cross–domain network service orchestration by introducing the X–MANO framework. X–MANO consists in a confidentially–preserving interface for inter–domain federation and in a set of abstractions (backed by a consistent information model) enabling network service life–cycle programmability. Said abstractions tackle all the aspect of *cross–domain* network service provisioning including on–boarding, scaling, and termination. We validate the proposed federation interface and information model by implementing them in a proof–of–concept X–MANO prototype and by using it to deploy a video transcoding network services in a multi–domain InP testbed. Finally, we release the proof–of–concept 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. Section III introduces the cross–domain orchestration challenges and the associated requirements. The X–MANO architecture and interfaces are described in Sec. IV. Finally, we draw our conclusions in Sec. V.

II. Related Work

In the deployment and orchestration of network services, one of the most important problem to take into account is the so–called virtual network embedding (VNE) problem. While the literature on single–domain VNE is already significant [7], works on multi–domain VNE have just started to appear [8], [9], [10], [11], [12]. Distributed approaches for VNE are provided in [8], [9]. However, they are more focused on

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

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

the theoretical analysis of the proposed protocols and not on architectural issues. In [10] the authors propose a method for abstracting domain resources. However, the proposed solution uses a centralized orchestrator requiring operators to rely on a 3rd party for cross-domain orchestration. Thus, the proposed solution is generally suitable for multi–technology orchestration (single operator) rather than for the multi–InP case. In [11] the authors propose a solution for the multi-domain VNE problem. However, they rely on a centralized broker model which hardly meets all possible use-cases. Indeed, operators will hardly agree to rely on a 3rd party entity unless a huge added value is forecast. Moreover, the problem formulation described in [11] is limited to the case of two InPs. In [12] authors propose a distributed multi–domain orchestration framework. Information about local infrastructures is kept within each domain. However, the framework forecast a specific domain orchestrator to be deployed which might constitute a high entry barrier for the solution.

Recent works can be found in literature investigating different architectural approaches for cross–domain orchestration [13], [14], [15], [16]. In [13] an inter– and intra–domain routing approach is investigated. However, the proposed solution relies on a centralized service controller which hardly meets the requirements identified for the multi-domain environment. In [14] the authors propose a reference framework for multi–domain orchestration coming from the work carried out within the H2020 5GEx project [17]. To the best of our knowledge, this is the first initiative to design a complete framework to achieve multi-domain orchestration. However, this work does not define an information model to abstract domains resources and to compose network services. Further, in [15] different multi–domain orchestration approaches are summarized: centralized, hierarchical, and cascading (or recursive). Recently, also ETSI released a report on architectural options taking into account multiple administrative domains [16].

The works above either address the multi–domain network service provisioning problem only from the VNE standpoint or, when tackling also the service management and orchestration aspects of the problem, support only one specific architecture, e.g. hierarchical, peer–to–peer, or recursive. Conversely in this work we set to defined a deployment–agnostic federation interface and information model for cross–domain network service orchestration.

III. Cross–domain Orchestration

In this section we introduce the challenges and requirements for cross-domain network service orchestration. Then in the next section we will describe how X–MANO addresses such requirements. Notice how throughout the paper we will use extensively the ETSI NFV terminology. For a detailed explanation of terms such as Virtual Network Function (VNF), VNF Descriptor (VNFD), Network Service (NS), and NS Descriptors (NSD) we refer the reader to [16], [18].

A. Business aspects and Architectural Considerations

Several architectures, each driven by different business requirements and use cases, can be used to enable multi–domain network service orchestration. The simplest approach is the *hierarchical* one [16] where different domains rely on a centralized orchestrator. From a business point–of–view this is a viable solution only for a single administrative domain as different operators will hardly provide global control of their infrastructures to a third party. Another approach is the *cascading* (or *recursive*) one where an operator exploits the network services exposed by another operator to serve its customers (e.g. a mobile network operator using a satellite operator for back–hauling). Finally, in the *peer–to–peer* model a network service is provided by pooling resources across several InPs possibly covering different geographical/technological domains. X–MANO supports all the use cases and architectural solutions described above by introducing a flexible, deployment–agnostic federation interface between heterogeneous administrative and technological domains.

B. Orchestration and Confidentiality

In the single–domain case, network service orchestration is performed assuming complete knowledge of the underlying resources. While this is still a valid assumption in the case of network services spanning across heterogeneous technological domains that belongs to the same operator, confidentially will be broken when multiple administrative domain are introduced. This highlights an important requirement for a multi–domain orchestration framework, i.e. the capability to hide operator specific details (e.g. traffic matrix). Similar considerations are made by the authors of [8], [9], [15]. X–MANO address this requirement by introducing an information model enabling each domain to advertise in a confidentially–preserving fashion capabilities, resources, and VNFs to an external entity. A Multi–Domain Network Service Descriptor (MDNS) implementing the aforementioned information model, allows network service developers to define network services without being exposed to the implementation details of the single domains.

C. Life–cycle Management

Irrespectively of the number of administrative and/or technological domains involved, VNFs and network services have specific life–cycle management requirements. For example, a video transcoding VNF may require a streaming VNF to be configured and running before it can start operating. Similarly an initialization script may require as input the output of other initialization scripts. As a result when VNFs belonging to the same network service are deployed across different domains it becomes harder to ensure consistent service on–boarding, scaling, and termination. This is due to the fact that different orchestrators must cooperate in order to deploy and operate a single network service. Notice how given the heterogeneity of all the possible scenarios and use cases, a unified cross–domain orchestration logic could prove either too rigid, if only a few hard–coded primitives are made available to network service developers, or too lax, if instead direct access to the single–domain orchestrator is provided. X–MANO addresses this requirement by introducing the concept of *programmable* network service which relies on a domain specific scripting language in order to allow network service developers to implement custom life–cycle management policies.

IV. X–MANO Overview

In this section we will introduce the main X–MANO components then we will describe in detail the X–MANO interfaces and the concept of programmable network service. It is worth stressing that the focus of this paper is to define the interfaces for cross–domain orchestration as a result the components described below are to be intended as logical leaving space for them to be implemented in different ways.

A. Components and Interfaces

The following components and interfaces can be identified:

- 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 Domain Northbound Interface (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, multi–tenancy, user management, and basic monitoring operations.
- Federation Manager (FM). An entity in charge of the cross–domain orchestration. The FM is essentially a cross–domain DOM. The FM exposes the Virtual Domain Interface (VD–If) which is essentially a D–If with support for the X–MANO information model. The FM supports the Federation Interface (F–If) which enables communication with the federated domains either via the FA (see below) or via direct connection to the DOMs, if they support the F–If communication protocol. Finally, the FM is in charge of splitting the MDNS into many single–domain NSD and to push them toward their relatives DOMs. In this manner a domain will only be aware about only a portion of the full network service.
- Federation Agent (FA). An entity bridging one or more DOMs with a FM. The FA is in charge of retrieving all the information related to VNFs and NSs available within one or multiple domains and of exposing them to the FM. The FA 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 federation interface, i.e. the F–If, while the later is the DOM northbound interface, i.e. the D–If. A FA can be embedded inside a DOM allowing the DOM to natively support the communication toward a FM.

The F–If interface allows a FM to hide the complexity/fragmentation of the underlying federated domains and let the upper entities to work over the them as if they were 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. Figure 1 illustrates several cross–domain orchestration architecture that are supported by the X–MANO interfaces. Notice how this include hierarchical, cascading, and peer–to–peer. In the latter case (peer–to–peer) the same FM behaves as master and slave at the same time, depending on request's origin point.

B. VNF Manifests

Resource advertisement is the way an FM is informed of the resources available at each federated domain. The advertisement is performed by each FA plugged to a federated domain. The way each FA retrieves such information is an implementation detail. Since we are focusing on VNFs and their compositions, the advertised resources will be the VNFs available in a given domain.

All the information advertised to the FM for a given VNF are provided in a manifest called VNF Manifest. A VNF Manifest contains the following information: (i) the VNF identifier and descriptor, (ii) a human readable description of the VNF, (iii) the set of functions that can be invoked on the VNF and the list of their expected parameters, and (iv) the set of parameters that can be monitored.

Notice how network services can be advertised by the FA as atomic VNFs. In particular a domain administrator can decide if some local network services shall be exposed to the FA. In this case, the FA will advertise each network service to the FM as a VNF with its own VNF Manifest. The FM will proceed advertising such a NS trough its VD–If as if the network services were VNFs hiding any information about their internal structure.

Domain administrators can advertises to the FM only a subset of the actions that can be triggered on VNFs, limit the connection points that available for chaining operations, constrain the set of VNF input parameters, and control the list of monitors that can be requested on a particular VNF. Notice how the translation between domain VNFs and VNF Manifest is currently done manually, nevertheless it is envisioned that such translation could be automated at the FA level using state–of–the–art model translation techniques.

C. Life–cycle Management

The module in charge of managing the network service life–cycle is the FM. This is done using a flexible programmable NSD which allows network service developers to customize the way network services are deployed and managed. NSDs have been extended with the scripts defining their life–cycle. Such scripts enable features like execution of operations in series or in parallel, storage of intermediate results in local variables (*Data Store*s) for later usage (across multiple domains), locking/unlocking of shared resources, and triggering of actions in response to a specific event or set of events.

NSD Scripts are interpreted by FM. In particular we can identify two main functions that are closely related with programmable NSD scripts: the *Trigger* function, and the *Handler* function (see Fig.2). The *Trigger* functions keeps track of a list of conditions defined in the NSDs, conversely the *Handler* function stores all the scripts that can be executed when one or more of the defined conditions is verified. It is worth noticing that *Triggers*, defined in the MDNS, are stored in the FM, since they are part of the multi–domain network service orchestration.

A notification system is in charge of detecting the events that can be part of a trigger (e.g., a change in the status of the network service, a new measurement, a given response from a previous operation, an error, etc.). The *Trigger* function

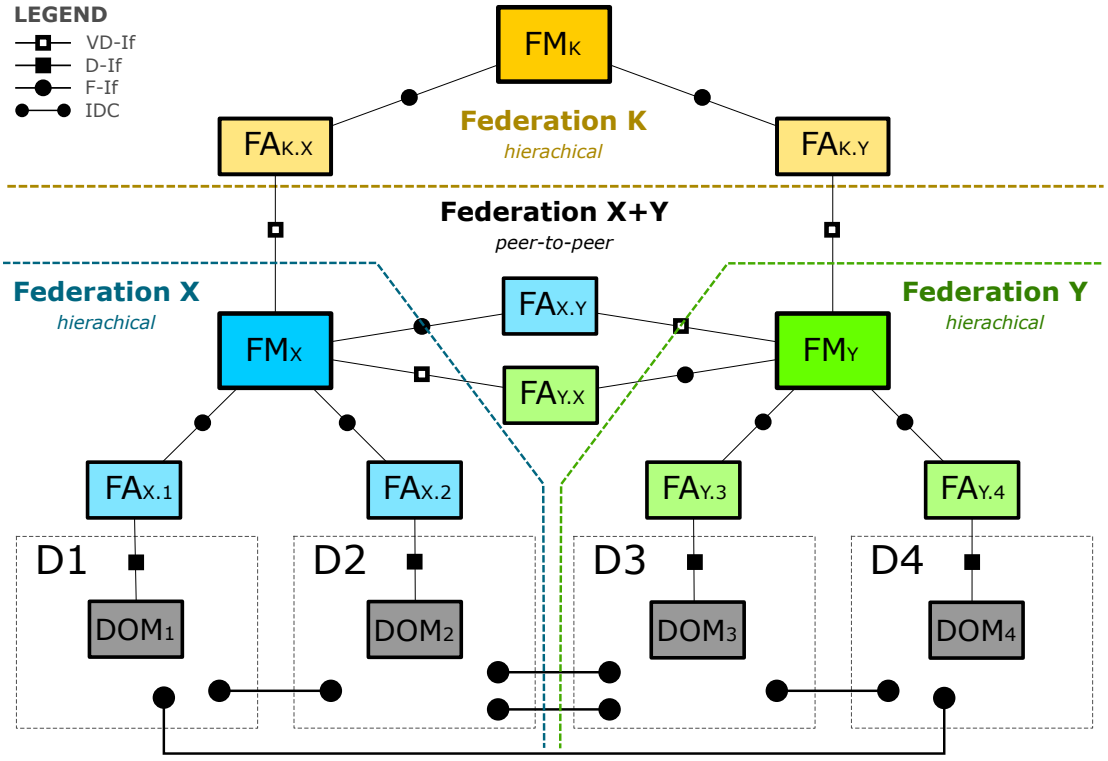

Fig. 1. Example of the different architectures supported by the X–MANO framework. Domains D1, D2, D3, and D4 have some Inter-Domain Connections (IDC) established between them. Each domain has its own DOM managing its local resources. D1 and D2 are federated by FM_X via Federation Agents $FA_{X.1}$ and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

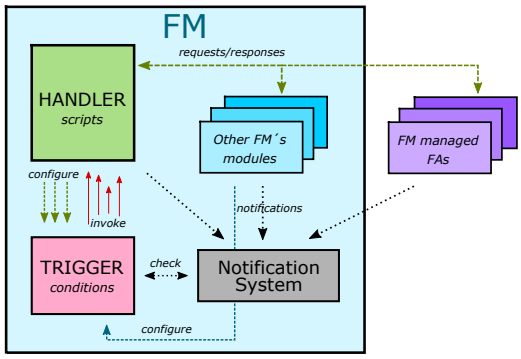

Fig. 2. Trigger-Handler mechanism.

continuously monitor the notification stream in order to check if one or more triggering conditions are met. When this happens, the *Trigger* looks up for the *Handler* that must be invoked and executes the associated script(s). While a script is running, new events (possibly generated by the same script) can be generated potentially triggering new handlers.

A MDNS can define multiple triggers declaring which operations have to be performed when certain conditions are met. A *Condition* specifies a comparison between two elements (the *left_item* and *right_item* fields) according to a logical comparison operator (the *operator* field). These two elements yield a type and a value. The type specifies the nature of the value (bool, int, double, string, etc.) while the value can be an hardcoded value or a *Data Store*. It is worth noticing that a *Trigger* can contain many "Condition" objects arranged in a hierarchical manner, where relations among these conditions are specified through logical operators.

*Data Store*s allows storing the results of an operation in one domain and use it as an argument for a another operation in different domain. Monitors, VNFs notification, and state change are all treated as *Data Store*s and can be accessed globally. *Data Store*s are particularly important because, when performing multi–domain orchestration it is often required to share information among VNFs located on different domains.

Tied to the conditions there are the operations, or *Steps*, that can be performed. Four types of *Step*s can be defined:

- *VNFs actions*. A list of VNF actions. For each VNF Action it is possible to provide a set of parameters and store the result of this operation in a *Data Store*. It is worth noticing that the actions that are defined in the same step are executed in parallel.
- *Elaborations*. This step allows to perform computations on *Data Store*s, e.g summing the value of two variables.
- *Conditional Step*. This step allows to check if a certain condition is valid and to react accordingly.
- *Lock/Unlock Domain*. This step is necessary whenever atomic operations have to be performed. This step provides exclusive access to the federated domain resources

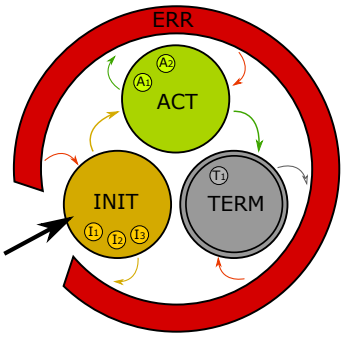

Fig. 3. X–MANO state machine with some user–defined internal states.

to a network service preventing collision with other network services that may run in parallel.

By the above description, it should be clear that the life–cycle of a network service can be totally managed by using user–defined NSD scripts. Nevertheless, a set of default states and the associated triggers/handlers is defined in order reduce the scripting burden for users that require only basic life–cycle management functions (see Fig. 3). Following these considerations, four main states can be identified:

- Initialization (INIT). Resources have been allocated and configured according to the network service requirements.
- Active (ACT). The network service has been fully instantiated and is ready to serve end–users.
- Termination (TERM). The network service has been terminated and all the resource have been released.
- Error phase (ERR). An un–handled exception occurred in the network service. The Error phase can be entered/exited from any of the other phases.

Starting from these four main phases, the network service developer can define other states and transition using the NSD scripting functionality. Depending on the number of VNFs and on the complexity of their interaction, the definition of the NSD script can become a complex task. However, we assume that such burden could be lighten by the adoption of a dedicated IDE providing a simplified graphical composition environment with auto–completion, debugger, and other tools for network service developers.

V. Conclusions

One of the main difference between single–domain and multi–domain network service orchestration is the level of awareness of the involved DOMs about the whole process. In the single–domain scenario the DOM has the whole situation under its control. Conversely, in the multi–domain scenario such global view is missing, since usually the different DOMs are not designed to interact with each other and to share information about the network service deployment process. This lack of synchronization among the involved DOMs can be fatal for the network service creation process, in particular when steps, which are supposed to be taken in a given order, are performed with no respect of the planned sequence.

The X–MANO framework proposed in this paper allows to coordinate the operations of different DOMs. This brings orchestration to the multi-domain level: indeed, the FM (the director) following the NSD (the score) coordinates and triggers the actions of the DOMs (the players). The combination and integration of the resource advertisement mechanism together with the flexibility in defining the NS life–cycle is the key of the proposed solution. A proof–of–concept implementation of the proposed framework has also been evaluated and released to the broader community under a permissive license.

As future work we can imagine a number of directions. The first is related to the automatic translation of domain VNF descriptors into VNF Manifest. Moreover also the decomposition of multi–domain NSDs into many single–domain NSD is also an challenging aspect. Likewise we also plan to formalize the programmable NSD scripting language definition and to embed it into a mainstream IDE such as Eclipse.

Acknowledgement

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

References

[1] IET, "Demand Attentive Network (DAN)," 2013.
[2] 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.
[3] T. Taleb, A. Ksentini, and R. Jantti, "Anything as a Service for 5G Mobile Systems," *IEEE Network*, vol. 30, no. 6, pp. 84–91, 2016.
[4] NGMN Alliance, "5G White Paper," Feb 2015.
[5] "OpenBaton." [Online]. Available: https://openbaton.github.io/
[6] "OPNFV." [Online]. Available: https://www.opnfv.org/
[7] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, 2013.
[8] 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.
[9] 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.
[10] I. Vaishnavi, R. Guerzoni, and R. Trivisonno, "Recursive, hierarchical embedding of virtual infrastructure in multi-domain substrates," in *Proc. of IEEE NetSoft*, London, UK, 2015.
[11] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011 – 1023, 2011.
[12] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Vertex-centric computation of service function chains in multi-domain networks," in *Proc. of IEEE NetSoft*, Seoul, South Korea, 2016.
[13] P. Iovanna, F. Ubaldi, F. Giurlanda, S. Noto, A. Priola, L. M. Contreras, V. Lopez, and J. P. F. P. Gimenez, "Effective elasticity for data centers interconnection in multi-domain wan: Information modelling and routing," in *Proc. of ECOC*, Valencia, Spain, 2015.
[14] R. Guerzoni, D. Perez-Caparros, P. Monti, G. Giuliani, J. Melian, R. Figueiredo, A. Ramos, C. J. Bernardos, G. Biczk, B. Sonkoly, F. Tusa, A. Galis, I. Vaishnavi, F. Ubaldi, A. Sgambelluri, C. Santana, and R. Szabo, "Multi-Domain Orchestration and Management of Software Defined Infrastructures: a Bottom-Up Approach," in *Proc. of EuCNC*, Athens, Greece, 2016.
[15] C. Bernardos, L. Contreras, and I. Vaishnavi, "Multi-domain network virtualization," Working Draft, Internet-Draft draft-bernardos-nfvrg-multidomain-01, October 2016.
[16] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*, Std. ETSI GS NFV-IFA 009, July 2016.
[17] H2020 5G Exchange (5GEx). [Online]. Available: http://www.5gex.eu
[18] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Architectural Framework*, Std. ETSI GS NFV 002, December 2014.