Need for a Transport API in 5G for Global Orchestration of Cloud and Networks Through a Virtualized Infrastructure Manager and Planner [Invited]

Arturo Mayoral, Raul Muñoz, Ricard Vilalta, Ramon Casellas, Ricardo Martínez, and Victor López

***Abstract*—The new 5G paradigm seeks for a scalable architecture that is able to efficiently manage the increasing volume of traffic generated by smart devices to be processed in a distributed cloud infrastructure. To this end, coordinated management of the network and the cloud resources forming an end-to-end system is of great importance. Software defined networking and network function virtualization architectures are the key enablers for integrating network and cloud resources, enabling cross optimization on both sides. This optimization requires efficient resource allocation algorithms, which take into account computing and network resources. In this paper, we propose an end-to-end orchestration architecture for distributed cloud and network resources aligned with the European Telecommunications Standards Institute management and orchestration architecture. The proposed architecture includes the virtual infrastructure manager and planner (VIMaP) component to enable dynamic resource allocation for interconnected virtual instances in distributed cloud locations. A heuristic algorithm for dynamic virtual machine graphs resource allocation is included to validate the VIMaP architecture and exploit its functionalities. Moreover, the control orchestration protocol is included between the architecture components to offer end-to-end transport services. Finally, the proposed architecture is experimentally validated, and the heuristic algorithm performance is evaluated.**

***Index Terms*—Cloud computing; Control orchestration protocol; Network function virtualization; Orchestration; Resource allocation; Software defined networking (SDN).**

I. Introduction

The fifth generation of mobile technology (5G) is not only about the development of a new radio interface but also of an end-to-end system. This end-to-end system includes the integration and convergence of all network segments (radio and fixed access, aggregation, metro, and core) with heterogeneous wireless and optical technologies together with massive cloud computing and storage infrastructures [1]. The 5G architecture shall accommodate a wide range of use cases with different requirements in terms of networking (e.g., security, latency, resiliency, bandwidth) and cloud resources [e.g., distributed nodes with cloud capabilities, edge/core data centers (DCs)]. Thus, from an administrative perspective, one of the main challenges for an infrastructure operator will be to provide multiple, highly flexible, end-to-end dedicated network and cloud infrastructure slices, over the same physical infrastructure, to different users or tenants in order to satisfy their application-specific requirements.

Software defined networking (SDN) has emerged as the most promising networking paradigm to realize the integration between cloud and network domains. In SDN, a centralized entity (commonly referred as a SDN controller) is able to program the forwarding behavior of the network elements through different southbound protocols (i.e., OpenFlow). Most of the SDN controllers also offer a northbound application programming interface (API) to higher level applications to expose its services (connectivity provisioning, topology discovery…). This abstraction enables network virtualization, that is, to slice the physical infrastructure and create multiple co-existing virtual tenant networks (VTNs) independent of the underlying transport technology and network protocols.

Often, to prevent vendor lock-in situations, telecommunication operators (Telcos) use different vendors' equipment in their transport networks. Specifically, in the optical networks, different vendor's equipment does not interoperate at the data plane level (only at the gray interface level) unlike regular Ethernet switches or IP routers. Moreover, each vendor offers its own control plane technology [e.g., SDN with some proprietary OpenFlow extensions or GMPLS and path computation element (PCE)] because of the need to configure vendor-proprietary parameters [e.g., forward error correction (FEC)], generating vendor islands. For these reasons, a single Telco's transport network may consist of multiple control domains, each controlled by a different entity, with its own northbound API.

SDN orchestration has been demonstrated as a feasible and scalable solution for multidomain, multitechnology network scenarios to provide end-to-end (E2E) network

Manuscript received June 23, 2016; revised August 26, 2016; accepted September 3, 2016; published October 12, 2016 (Doc. ID 269074).

A. Mayoral (e-mail: arturo.mayoral@cttc.es), R. Muñoz, R. Vilalta, R. Casellas, and R. Martínez are with CTTC, Castelldefels, Spain.

V. López is with Telefónica I+D/Global CTO, Madrid, Spain.

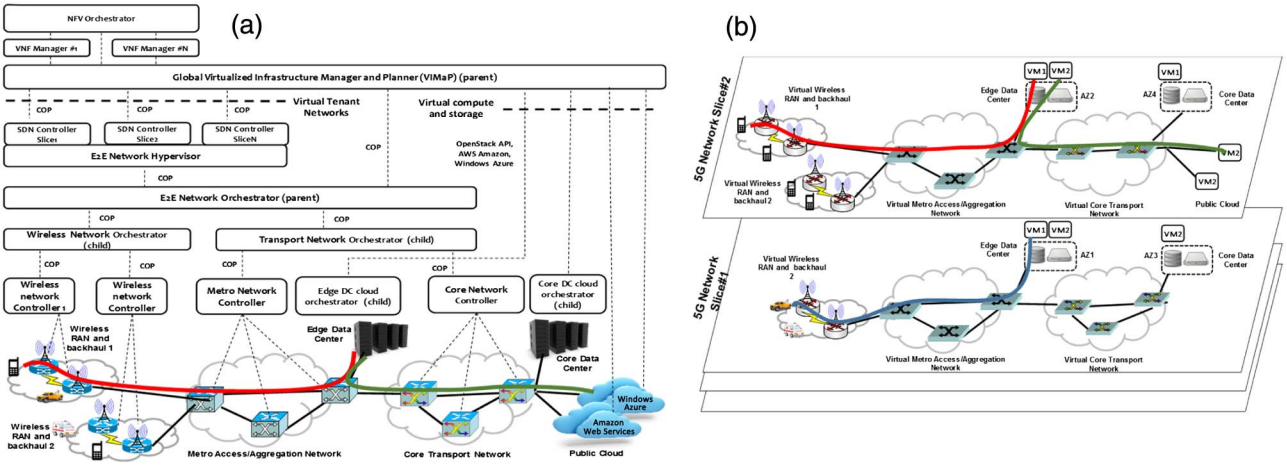

Fig. 1. (a) 5G network architecture for dynamic provisioning of virtual computation, storage, and networking across distributed cloud infrastructures and heterogeneous networks. (b) 5G network slices supporting different cloud and network requirements.

services in [2] and [3]. A multidomain SDN network orchestrator acting as a unified transport network operating system (or controller of controllers) allows the control (e.g., E2E transport service provisioning), at a higher abstracted level of heterogeneous network technologies regardless of the specific control plane technology employed in each domain (e.g., SDN/OpenFlow or GMPLS/PCE). In this line, the application-based network operations (ABNO) [4], promoted by IETF, provides a reference guideline for the development of a multidomain SDN orchestrator architecture.

In this context, we propose the control orchestration protocol (COP) [5] as a unified interface for SDN orchestration in multidomain transport networks. COP was developed within the STRAUSS project (http://www.ict-strauss.eu) as the first transport API to provide a common interface, which abstracts the particular control plane technology of a given transport domain. COP is a research-oriented initiative, developed with the aim of stressing the need of a common transport API in SDN. In [6], the interoperability between different SDN controllers and orchestrators using COP was experimentally demonstrated in a multipartner testbed. The COP definition is open for discussion and can be downloaded and contributed at https://github.com/ict-strauss/COP. COP standardization efforts were merged into the latest OIF/ONF Transport SDN API [7], which is clearly aligned with the COP objectives, to extend the scope of our proposal toward its adoption by vendors and operators in their solutions.

The COP and SDN orchestration are the key enablers on which we build our proposed end-to-end architecture for distributed cloud and network orchestration [see Fig. 1(a)]. On top of the architecture, the global virtualized infrastructure manager and planner (VIMaP) is responsible for the global orchestration of distributed cloud controllers and the transport network infrastructure. The VIMaP implements the same functionalities of the VIM, as defined by the European Telecommunications Standards Institute in the NFV management and orchestration reference architecture [8], i.e., it is responsible for the NFV infrastructure resources management, such as computation, storage, and networking. Our proposal extends the VIM with a new planner component (VIMaP) for resource optimization and planning, which provides an online platform to run resource allocation algorithms for the arriving infrastructure slice requests from the different tenant applications.

Thus, our contribution in this paper is twofold. On the one hand, we have extended the work in [9], where the proposed SDN hierarchical architecture using the COP as a unified interface has been complemented by the introduction of the VIMaP. On the other hand, the second contribution consists of modeling the resource allocation problem of dynamically provisioning of infrastructure-as-a-service (IaaS), which in this paper we refer to as the virtual machine graphs (VMGs) provisioning problem. We also include a heuristic baseline solution based on the Greedy approach for the selection of DCs and first fit (FF) for the VM allocation. The objective is to provide a baseline implementation of a resource allocation algorithm for the latter experimental validation of the whole architecture (with a special focus on the VIMaP component and the VMG allocation). However, to verify its validity, the proposed algorithm performance is evaluated and compared with a random fit based solution in a controlled simulation scenario.

This paper is organized as follows: Section II includes the proposed architecture description and the COP's description. Section III focuses on the novelties of the VIMaP. In Section IV, the VMG allocation problem and the proposed heuristic are formally presented; moreover, the section is completed with the simulation environment and the results. Finally, in Section V, an experimental validation of the architecture is presented.

II. Proposed Network Architecture for Distributed Cloud and Heterogeneous Network Orchestration

The considered network scenario is composed of multiple wireless radio access and backhaul technologies

and multidomain, multilayer, and multivendor transport networks, with heterogeneous control domains, interconnecting distributed cloud infrastructures (both private and public). The use of COP between the SDN network orchestrator and control layers allows the simplification and optimization, in terms of scalability and compatibility, between the different modules, which compose the SDN architecture. COP unifies all the orchestration functionalities into a single protocol paradigm. The COP information model is described in YANG modeling language, with REST conf as transport protocol using JavaScript Object Notation (JSON) encoding for data transmission. In brief, COP is composed of three main base functions:

1) Topology service: This provides topological information about the network, which includes a common and homogeneous definition of the network topologies included in the TE databases of the different control instances.
2) Path computation service: This provides an interface to request and return path objects that contain the information about the route between two endpoints.
3) Call service: This is based on the concept of call/connection separation and provides a common provisioning model, which defines an end-to-end connectivity provisioning service.

One benefit of this architecture resides in the ability to perform unified control and management tasks (e.g., E2E provisioning services) of different radio access and transport network technologies by means of the same SDN network orchestrator. However, for scalability, modularity, and security purposes, it may be desired to consider a hierarchical orchestration approach with different levels of hierarchy (parent/child architecture). Each higher level has the potential for greater abstraction and broader scope (e.g., we may consider one orchestrator for the RANs and another for the transport networks), and each level may exist in a different trust domain. The level interface might be used as a standard reference point for inter-domain security enforcement. In our approach, the COP can be used as the northbound interface (NBI) of the child SDN orchestrator and as the southbound interface (SBI) of a parent SDN orchestrator in order to provision E2E services. A parent/child SDN orchestrator architecture based on ABNO has been previously validated for E2E multilayer (packet/optical) and multidomain transport provisioning across heterogeneous control domains (SDN/OF and GMPLS/AS-PCE) employing dynamic domain abstraction based on virtual node aggregation in [10].

In the proposed system architecture [Fig. 1(a)], a network hypervisor is placed on top of the E2E network orchestrator. It is responsible for partitioning and/or aggregating the abstracted resources provided by the E2E network orchestrator into virtual resources, interconnecting them to compose multiple end-to-end virtual tenant networks (VTNs) with different VTN topologies while sharing the same physical infrastructure. It is also responsible for representing an abstracted topology of each VTN (i.e., network discovery) to a tenant SDN controller and for it to remotely control the virtual network resources (i.e., dynamic provisioning, modification and deletion of connections) allocated to their corresponding VTN, as if they were real resources, through a well-defined interface (e.g., OpenFlow protocol, or the COP). The network hypervisor can dynamically create, modify, and delete VTNs in response to application demands (e.g., through a traffic demand matrix describing resource requirements and QoS for each pair of connections). The proposed multidomain network hypervisor architecture has been proposed and assessed in [11].

Virtualization of computation, storage, and networking resources in DCs is provided by private clouds through distributed cloud orchestrators (children), which may be deployed with different software distributions (e.g., OpenStack, OpenNebula) or by public cloud. Each cloud orchestrator enables us to segregate the DC into availability zones for different tenants and instantiate the creation/migration/deletion of virtual machine (VM) instances (computing service), storage of disk images (image service), and the management of the VM's network interfaces and the intra-DC network connectivity (networking service). On the other hand, the global VIMaP (parent) targets the global management of the virtual computation, storage, and networking resources for the different slices provided by the tenant SDN controllers and the distributed cloud orchestrators. It acts as a unified cloud and network operating system providing, for each slice, the dynamic and global provision, migration, and deletion of VMs and the required end-to-end connectivity between the distributed virtual cloud infrastructures across the corresponding multilayer VTN [Fig. 1(b)]. A key enabler of such an integration is the COP, which is used as a NBI by the tenant SDN controllers, providing a common control of the VTNs. A preliminary architecture of a global cloud and network orchestrator named SDN IT and network orchestrator (SINO) has been defined and evaluated in [12] and [13].

III. Virtual Infrastructure Manager and Planner

In this section, we present the VIMaP architecture, including the description of its main building blocks, which are shown in Fig. 2. The VIMaP has been designed to provide coordinated orchestration of network and cloud resources distributed among different cloud providers and locations. The VIMaP provides per-tenant programmability of its own dedicated resources, it performs the partitioning of the underlying infrastructure, exposing an abstracted view of virtual infrastructure slices to each tenant.

Initially, the VIMaP is requested to provide a virtual infrastructure slice to a dedicated tenant. This request includes a set of virtual instances interconnected forming a VMG. The VIMaP architecture includes a planner component dedicated to performing resource planning optimization. Different resource optimization policies may be applied depending on the tenant and provider requirements. In [14], the authors assessed the problem of VMG resource allocation in distributed DC scenarios by finding the minimum diameter graph (in terms of distance) to minimize the latency between VMs. The authors of [15]

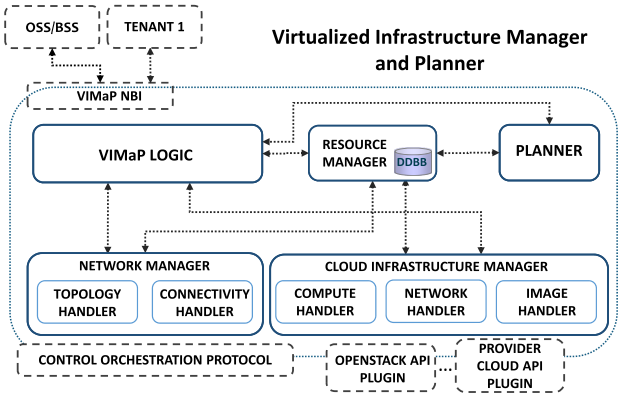

Fig. 2. VIMaP internal architecture, building blocks.

propose a resource allocation approach taking into account the distance between the DC and the network load to select the connection path. The VIMaP architecture allows the VIMaP LOGIC component to select the preferred algorithm depending on the desired resource allocation policy. The algorithm receives the resource allocation requests from the VIMaP logic, and it obtains all the substrate infrastructure information from the resource manager component, which maintains up-to-date information of both the cloud and the network underlying infrastructure.

The VIMaP includes a dedicated configuration interface for slice provisioning, which is exposed to OSS/NMS management systems through a RESTful API. The VIMaP LOGIC component is responsible for orchestrating the workflows among the different architectural components in order to provision the cloud and network resources for an upcoming request. It is responsible for performing context-aware orchestration, exposing to each tenant only those resources allocated to the tenant by means of virtual representation. It includes an NBI, which exposes the custom set of VIMaP programmable resources to each tenant.

The resource manager is responsible for storing and maintaining the up-to-date state of all virtual and physical sources controlled by the VIMaP. It is also responsible for maintaining the resource allocation relationship between the requested virtual resources and the allocated physical resources.

Network manager functions are twofold: First they provide the southbound interface toward network infrastructure controllers, including the necessary API or protocol implementations. As we have presented before, the COP is the protocol chosen to unify the network orchestration interface toward different SDN controllers. Second, the network manager is responsible for managing the virtual network resources of each tenant. The network manager correlates the VTN representation with the dedicated SDN controller slice, there is a 1:1 relation between a VTN and a SDN controller slice.

The cloud infrastructure manager is responsible for distributed cloud orchestration. As opposed to the network manager, it is responsible for the partitioning and aggregation of cloud resources, which might be distributed across different clouds (private, public). Once the selected DCs are allocated for a given tenant, it is responsible for creating a tenant session on each child cloud system and mapping all these client sessions to the corresponding VIMaP *TenantID*. Once this initial abstraction is performed, it is responsible for aggregating all the resources distributed among different clouds into a single unified view accessible by the tenant through the VIMaP NBI. This is performed by populating the resource manager database with virtual representation of the resources deployed in the underlying infrastructure; these resources are segmented by its corresponding VIMaP global *TenantID*.

IV. Virtual Machine Graph Allocation

In this section, we first describe the general VMG allocation problem. Then, we present a reduction of the problem based on constructing the aggregated VMG solution graph, where the objective is to find groups of VMs to be allocated together in the same substrate hosting nodes. This reduction is modeled based on a constrained mapping function. Finally, a heuristic algorithm solution to this problem is proposed, and simulation results for the algorithm behavior are provided.

A. Virtual Machine Graph Allocation Problem Definition

Substrate infrastructure. We model the substrate infrastructure as a directed graph and denote it by $G^S = (N^S, H^S, L^S)$, where N^S is the set of substrate switching nodes, H^S is the set of substrate hosting nodes (DCs), and L^S denotes the set of substrate links $l^s = (u, v)$, $l^s \in L^S$, $\forall u, v \in N^S \cup H^S$.

Virtual machine graph request. We denote by a directed graph $G^S = (H^V, L^V)$ the VMGP request. H^V denotes the set of virtual hosts (VMs) and L^V denotes the set of links between virtual hosts.

Now we define a set of capacity functions for the substrate and virtual resources. Each host (physical or virtual) $h^x \in H^x, x \in \{S, V\}$ is attributed with a set of A attributes whose capacities are denoted as $c_a(h^x)$, $a \in A$, $h^x \in H^X$, $A \in \{\text{CPU}, \text{MEM}, \text{STO}\}$ (we consider only CPU, memory and storage as host attributes). Moreover, each link $l^x \in L^X$ is associated with a bandwidth capacity $bw(l^x)$. We also denote P^S as the set of free loop paths in the substrate network between hosting nodes.

The objective is to find a mapping function for all virtual hosts and links to the substrate infrastructure as

$$M : (H^V, L^V) \mapsto (H^S, P^S).$$

In the next subsection, a reduction of the problem is proposed, and the constraints in terms of capacities for hosts and links are introduced.

B. VMG Mapping Problem

To solve the above-described problem, we propose a first reduction, which consists of (1) finding a VM allocation among the substrate hosting nodes, and (2) finding a feasible allocation solution for the links connecting VM in different hosting nodes. It is assumed that several virtual hosts can be placed in the same substrate hosting node if enough computing resources are available in the substrate node for the aggregated capacity of the virtual hosts allocated to it.

We model the aggregated VMG solution graph as $G' = (H', L')$, where each $h' \in H'$ denotes a subset $h' \subseteq H^V$ of virtual hosts. Given the power set of all possible subsets of H^V denoted as $\mathcal{P}(H^V)$, the subsets included in a hosting allocation solution $H' \subset \mathcal{P}(H^V)$ must be complementary and disjoint, i.e., which satisfy both $\bigcup_{H'} h' = H^V$ and $\bigcap_{H'} h' = \emptyset$.

On the other side, L' denotes the set of links between virtual hosts in different aggregated subsets $l' = (u, v)$, $\forall l' \in L'$, $u \in h'_i$, $v \in h'_j$, and $h'_i \neq h'_j$.

Once G' has been described, we can define the mapping function between the VMG solution graph and the substrate infrastructure as

$$M : (H', L') \mapsto (H^{S'}, P^{S'}),$$

where $H^{S'} \subseteq H^S$, $P^{S'} \subseteq P^S$. The mapping function can be split into hosting and link mapping as

- Hosting mapping function:

$$M^H : (H') \mapsto (H^{S'}),$$

which satisfies

$$\forall\, h' \in H', \forall\, h^{s'} \in H^{S'}, \sum_{\forall\, h^v \in h'} c_a(h^v) \leq c_a(h^{s'}). \quad (1)$$

In order to compare the sizes of the hosts (physical or virtual) in relative terms, we define the function weight, as the weighted sum of the individual computing capacities, we use the constants α, β, and γ to weight up the CPU, memory, and storage capacities, respectively:

$$\text{weight}(h^x) = \alpha c_{\text{CPU}}(h^x) + \beta c_{\text{MEM}}(h^x) + \gamma c_{\text{STO}}(h^x). \quad (2)$$

- Link mapping function:

$$M^L : (L') \mapsto (P^{S'})m,$$

which satisfies

$$\forall\, l' \in L', \forall\, p^{s'} \in P^{S'},\ \ bw(l') \leq BW(p^{s'}),$$
$$\text{where } BW(p^s) = \min_{\forall l^s \in p^{s'}} bw(l^s). \quad (3)$$

C. Baseline VMG Embedding Algorithm

The problem has been reduced to find a feasible allocation for the solution graph G', which satisfies Eqs. (1) and (3).

We assess the problem in two steps:

- 1. Following a Greedy procedure, we select the minimum number of substrate hosting nodes with enough capacity to allocate all the virtual hosts in H^V, which are embedded sequentially following a FF approach (see Algorithm 1).
- 2. Based on the selected $H^{s'} \subseteq H^S$, we employ the constrained shortest path first (CSPF) algorithm to find a feasible path in the substrate network, for each s-t pair allocated in different substrate hosting nodes (see Algorithm 2).

Algorithm 1: Greedy FF Host Mapping (H^S, H^V)

Input: H^S: Substrate hosting nodes, H^V: Virtual hosts
Output: $H', H^{S'}$: host solution set

```
Sort H^S = h_1^s, h_2^s, ..., h_n^s in decreasing order by its weight
H^{S'} ← Ø
H' ← Ø
H^{v'} ← H^V
while Σ_{∀ h^{s'} ∈ H^{S'}} (c_a(h^{s'})) < Σ_{∀ h^v ∈ H^V} (c_a(h^v)),
  ∀ a ∈ A do
  h^s ← H^S.pop()
  currentC_a ← c_a(h^s), ∀ a ∈ A
  current_s ← Ø
  for v in H^{v'} do
      If one of currentC_a < c_a(v), ∀ a ∈ A then
          H^{v'} ← H^{v'} − current_s
          break
      else
          current_s ← current_s ∪ {v}
          currentC_a ← currentC_a − c_a(v), ∀ a ∈ A
      end if
  end for
  H' ← H' ∪ current_s
  H^{s'} ← H^{s'} ∪ h^s
end while
return M^H: H' ↦ H^{S'}
```

Algorithm 1 first computes the Greedy and the FF host mapping procedure to find the minimum cluster with enough capacity to allocate virtual hosts within the VMG request. First, it sorts the substrate host set in decreasing order by weight, and it sequentially allocates the virtual hosts into the substrate hosting nodes with higher capacities. As a result, this function returns the solution subset with minimum size $H^{S'} \subseteq H^S$.

Algorithm 2: CSPF Link Mapping ($H', H^{S'}, L^S, L^V$)

Input: $H', H^{S'}$: substrate host solution set,
L^S: Input substrate links,
L^V: Input links request
Output: $M : (H', H^{S'}), (L', P^{S'})$: Mapping solution from $G^V \mapsto G^S$

```
for (u, v) in L^V do
      if h'_u ≠ h'_v then
        L' ← L' ∪ (u, v)
      end if
end for
```

for $l'(u,v)$ **in** L' **do**
  $p^{s'} \leftarrow CSPF(G^S, h_u^{s'}, h_v^{s'}, bw(l'))$
end for
return $M : (H', L') \mapsto (H^{S'}, P^{S'})$

Algorithm 2 receives the host solution subset and both substrate and virtual links of the VMG request. Based on the host mapping solution, for each virtual link $l'(u,v)$, a feasible path p' between nodes allocated to different $h'_u, h'_v, i \neq j$ is calculated. We use the CSPF algorithm with the $bw(u,v)$ as a constraint parameter. If there is a feasible path for each $l' \in L'$, the mapping solution is returned: $(H', L') \mapsto (H^{S'}, P')$.

D. VMG Allocation Results

In this section, we present the evaluation of the proposed heuristic baseline solution comparing with a random fit based algorithm. The random solution differs on the DC selection strategy but keeps CSPF to ensure path feasibility in the virtual link selection stage.

The substrate infrastructure scenario employed for the experiments is an extended version of the NSFNET of 14 nodes and 42 unidirectional links and six DCs (Fig. 3). For simplicity, the DCs are co-located within the same network node locations, and the connectivity between a DC and its corresponding network nodes is modeled to have infinite bandwidth. The substrate infrastructure is initially configured with predefined capacities, which are maintained along all the experiments. The values of the capacities of each DC are uniformly distributed among the values included in each range, as depicted in Table I.

In the VMG requests, the number of virtual nodes is randomly determined by a uniform distribution between 2 and 20. Each pair of nodes is randomly connected with probability of 0.5; in total, we will have $n(n-1)/4$ links on average. The capacities of the virtual hosts and the virtual links are also selected randomly following a uniform distribution along the values depicted in Table I.

The VMG requests arrive at the VIMaP following a Poisson process on which the arrival rate is varying. The holding time of the VMG requests in the system follows

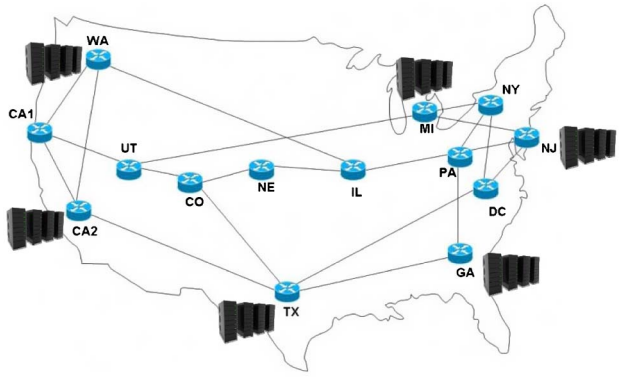

Fig. 3. NSF Network of 14 nodes with 6 DCs.

TABLE I
EXPERIMENT PARAMETER CONFIGURATION

Parameter	Values
H^S CPU values	[100, 200, 400]
H^S memory values	[200, 400, 800]
H^S storage values	[10,000, 20,000, 40,000]
L^S bandwidth	100 Gbps
H^V CPU values	[1,2,4,8]
H^V memory values	[2,4,8,16]
H^V storage values	[20, 40, 80, 160]
L^V bandwidth	(0.1:1) Gbps

an exponential distribution with 10 time windows on average. We run all the simulations for 10,000 requests for each instance of the simulation.

The results of the simulation for different loads can be seen in Fig. 4. The results show a slightly better performance of the Greedy FF approach compared with the random. This result is explained by the fact that the Greedy approach minimizes the number of DCs selected in the first stage, minimizing as well the number of connections between DCs and thus decreasing network utilization. The differences obtained are minimal, showing that the dominant factor for the blocking probability in this experiment is not the exhaustion of network resources but the cloud. In this paper, the target is to present the problem of VMG allocation and the baseline solution for the proposed VIMaP architecture. It is intended for future work in the evaluation of more complex algorithms and its comparison within the VIMaP.

V. EXPERIMENTAL VALIDATION

The proposed architecture has been validated in the cloud computing platform and transport network of the ADRENALINE testbed. The cloud computing platform is

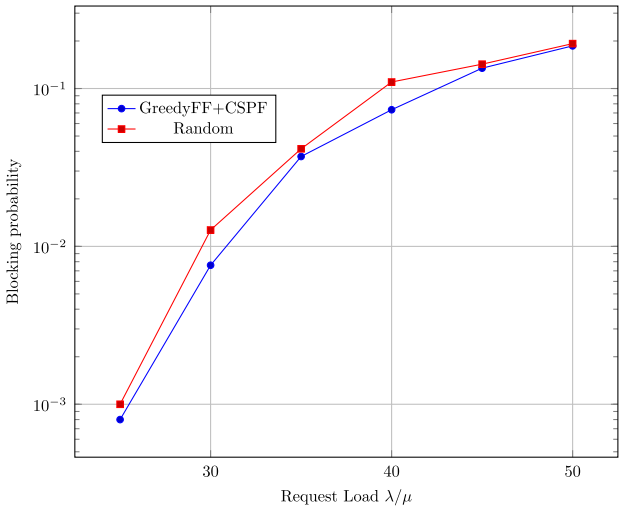

Fig. 4. VMG request blocking probability of GreedyFF+CSPF and random Fit+CSPF algorithms.

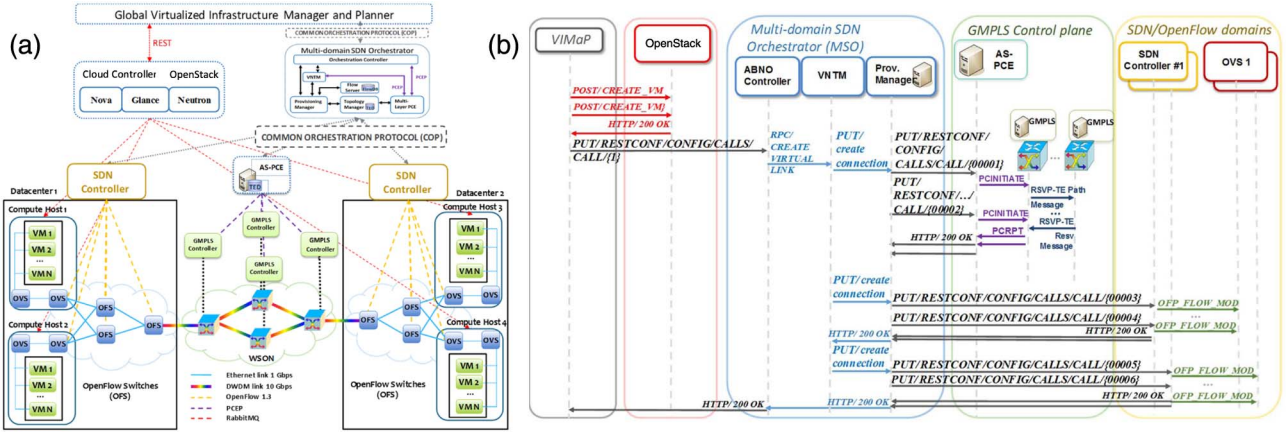

Fig. 5. (a) Experimental scenario for DC interconnection. (b) Integrated IT/SDN orchestration workflow.

controlled using OpenStack (Havana release), which has been deployed into servers with 2× Intel Xeon E5-2420 and 32 GB RAM each. An OpenStack controller node and four compute nodes have been set up in different network locations. Each DC network is composed of four OpenFlow switches deployed on COTS hardware and using OpenVSwitch (OVS) technology. Two hybrid packet/optical aggregation switches based on OVS as well and with a 10 Gb/s XFP tunable transponder connecting to the DWDM network as alien wavelengths. Finally, the GMPLS/PCE-controlled optical network is composed of an all-optical WSON with two ROADMs and two OXCs. The multidomain SDN orchestrator (MSO) and VIMaP entities have been mostly implemented in Python with the exception of the multilayer PCE, which has been implemented in C++ [16].

The COP has been employed as a transport API for the orchestration of two SDN OpenDaylight Helium controllers responsible for controlling the Ethernet intra-DC domains via OpenFlow 1.3, and the optical transport network via an AS-PCE with instantiation capabilities as a single interfacing point for the GMPLS control plane. In the experimental validation, we have introduced COP agents on top of SDN controllers in order to translate the received COP commands to SDN controller NBIs. Figure 5(a) shows a multidomain network scenario where two geographically distributed DCs are interconnected through the WSON. Figure 5(b) illustrates the integrated IT/SDN orchestration workflow for the on-demand deployment of two VMs in the cloud (one on each DC location) and the E2E connectivity provisioning across the proposed scenario. The network orchestration is performed using the proposed COP between the SINO-MSO and, consequently, between the MSO and the per-domain controllers. For this experimental validation, a bidirectional CALL_SERVICE is requested by the SINO to provide an E2E connectivity to the previously deployed VMs. The MSO first requests the creation of a virtual link in the upper layer topology (L2), which is translated internally by the VNTM MSO module into two unidirectional L0 CALL_SERVICEs sent to the AS-PCE through the provisioning manager. They trigger, in the AS-PCE, the creation of the corresponding GMPLS connections (label switched paths [LSPs]). Afterward, the provisioning of the E2E service in the upper layer is requested to the SDN controllers, by two new unidirectional CALL_SERVICESs to each domain.

The traffic capture shown in Fig. 6 validates the use of the COP. First, we can observe the request for virtual machine (VM) creation from VIMaP toward the cloud controller (which is running on the same server). The creation time for a single VM is 15 s, which includes the necessary time to boot up the VM. Second, in Fig. 7 we can observe the call request (Call Id: 1) from the VIMaP toward the

Time	Source	Destination	Info
REF	VIMAP-ABNO	VIMAP-ABNO	POST /create_vm HTTP/1.1 (application/json)
16.178267	VIMAP-ABNO	VIMAP-ABNO	HTTP/1.1 200 OK (text/html)
16.181848	VIMAP-ABNO	VIMAP-ABNO	POST /create_vm HTTP/1.1 (application/json)
30.914099	VIMAP-ABNO	VIMAP-ABNO	HTTP/1.1 200 OK (text/html)
REF	VIMAP-ABNO	VIMAP-ABNO	POST /restconf/config/calls/call/1 HTTP/1.1 (ap
0.123129	VIMAP-ABNO	SDN-CTL-1	POST /restconf/config/calls/call/00001 HTTP/1.1
0.287926	SDN-CTL-1	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
0.329396	VIMAP-ABNO	AS-PCE	POST /restconf/config/calls/call/00002 HTTP/1.1
1.439163	AS-PCE	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
1.493375	VIMAP-ABNO	SDN-CTL-2	POST /restconf/config/calls/call/00003 HTTP/1.1
1.527963	SDN-CTL-2	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
1.628074	VIMAP-ABNO	SDN-CTL-2	POST /restconf/config/calls/call/00004 HTTP/1.1
1.660056	SDN-CTL-2	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
1.699451	VIMAP-ABNO	AS-PCE	POST /restconf/config/calls/call/00005 HTTP/1.1
2.308023	AS-PCE	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
2.358390	VIMAP-ABNO	SDN-CTL-1	POST /restconf/config/calls/call/00006 HTTP/1.1
2.502622	SDN-CTL-1	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)
2.519650	VIMAP-ABNO	VIMAP-ABNO	HTTP/1.1 200 OK (application/json)

Fig. 6. Wireshark network traces capture.

JavaScript Object Notation: application/json
Object
Member Key: "trafficParams"
Member Key: "callId"
Member Key: "zEnd"
Object
Member Key: "routerId"
Member Key: "interfaceId"
Member Key: "endpointId"
String value: 77:77:77:77:77:77:77:03_3
Member Key: "aEnd"
Object
Member Key: "routerId"
Member Key: "interfaceId"
Member Key: "endpointId"
String value: 77:77:77:77:77:77:77:01_1
Member Key: "transportLayer"
Member Key: "match"
Object
Member Key: "ethDst"
String value: 00:14:f5:ce:9e:13
Member Key: "ethSrc"
String value: 00:15:3f:5d:01:6c

Fig. 7. Example of JSON COP Call object.

multidomain SDN orchestrator (based on ABNO). In the call service request, several constraints can be observed, such as the requested end points (aEnd, zEnd), several traffic parameters (such as requested bandwidth), the requested transport layer, and the MAC addresses of the interconnected VMs. The ABNO computes the necessary domain call requests and sends them toward the AS-PCE for the optical domain (Call Id: 00002, 00005), SDN Controller 1 (Call Id: 00001, 00006), and SDN Controller 2 (Call Id: 00003, 00004). The multidomain call service setup delay is of 2.52 s.

VI. Conclusion and Future Work

The definition of a transport API that abstracts a set of control plane functions used by a SDN controller, allowing the SDN orchestrator to uniformly interact with heterogeneous control domains, will pave the way toward the required transport network interoperability as well as integration with wireless networks and cloud infrastructure. In this paper we have presented the control orchestration protocol and experimentally validate its utility for cloud and network orchestration. Moreover, we have defined and experimentally demonstrated the novel VIMaP component for the resource allocation and planning of VMGs. We have formally modeled the problem and included a baseline heuristic solution to evaluate the proposed architecture.

For the next steps in this research, we expect to extend the work done for the VMG problem definition for the development of more complex algorithms, which exploit cross-resource optimization of cloud and network domains.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 671598 (5G-Crosshaul) and the Spanish MINECO project DESTELLO (TEC2015-69256-R).

References

[1] NGNM Aliance, "5g white paper," 2015.

[2] R. Muñoz, R. Vilalta, R. Casellas, R. Martnez, F. Francois, M. Channegowda, A. Hammad, S. Peng, R. Nejabati, D. Simeonidou, N. Yoshikane, T. Tsuritani, V. López, and A. Autenrieth, "Transport network orchestration for end-to-end multilayer provisioning across heterogeneous SDN/openflow and GMPLS/PCE control domains," *J. Lightwave Technol.*, vol. 33, pp. 1540–1548, 2015.

[3] Y. Yoshida, A. Maruta, K.-I. Kitayama, M. Nishihara, T. Tanaka, T. Takahara, J. C. Rasmussen, N. Yoshikane, T. Tsuritani, I. Morita, S. Yan, Y. Shu, Y. Yan, R. Nejabati, G. Zervas, D. Simeonidou, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, A. Aguado, V. López, and J. Marhuenda, "SDN-based network orchestration of variable-capacity optical packet switching network over programmable flexi-grid elastic optical path network," *J. Lightwave Technol.*, vol. 33, pp. 609–617, 2015.

[4] D. King and A. Farrel, "A PCE-based architecture for application-based network operations," IETF RFC 7491, 2015.

[5] V. Lopez, L. Miguel, J. Foster, H. Silva, L. Blair, J. Marsella, T. Szyrkowiec, A. Autenrieth, C. Liou, A. Sadasivarao, S. Syed, J. Sunjun, B. Rao, and F. Zhang, "Demonstration of SDN orchestration in optical multi-vendor scenarios," in *Optical Fiber Communication Conf.*, 2015, paper Th2A-41.

[6] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martinez, M. S. Moreolo, J. M. Fabrega, S. Yan, A. Aguado, E. Hugues-Salas, S. Peng, F. Meng, Y. Shu, G. Zervas, R. Nejabati, D. Simeonidou, J. M. Gran, V. López, O. Gonzalez de Dios, J. P. Fernández-Palacios, P. Kaczmarek, R. Szwedowski, T. Szyrkowiec, A. Autenrieth, N. Yoshikane, X. Cao, T. Tsuritani, I. Morita, M. Shiraiwa, N. Wada, M. Nichihara, T. Tanaka, T. Takahara, J. C. Rasmussen, Y. Yoshida, and K. Kitayama, "First experimental demonstration of a distributed cloud and heterogeneous network orchestration with a common transport API for E2E services with QoS," in *Optical Fiber Communication Conf.*, Mar. 2016, paper Th1A-2.

[7] OIF-ONF, "Global transport SDN prototype demonstration," White paper, 2014.

[8] "Network function virtualization (NFV): Architectural framework," ETSI GS NFV 002 v.1.1.1, 2013.

[9] R. Muñoz, A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, and V. López, "The need for a transport API in 5G networks: The control orchestration protocol," in *Optical Fiber Communication Conf.*, 2016.

[10] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martinez, "Hierarchical SDN orchestration for multi-technology multi-domain networks with hierarchical ABNO," in *European Conf. on Optical Communication (ECOC)*, 2015, pp. 1–3.

[11] R. Vilalta, R. Muñoz, R. Casellas, R. Martinez, S. Peng, R. Nejabati, D. Simeonidou, N. Yoshikane, T. Tsuritani, I. Morita, V. Lopez, T. Szyrkowiec, and A. Autenrieth, "Multidomain network hypervisor for abstraction and control of OpenFlow-enabled multitenant multitechnology transport networks [Invited]," *J. Opt. Commun. Netw.*, vol. 7, pp. B55–B61, 2015.

[12] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, and R. Martinez, "Experimental seamless virtual machine migration using a SDN IT and network orchestrator," in *Optical Fiber Communication Conf. (OFC)*, Mar. 2015.

[13] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martinez, and J. Vilchez, "Integrated IT and network orchestration using OpenStack, OpenDaylight and active stateful PCE for intra and inter data center connectivity," in *European Conf. on Optical Communication (ECOC)*, 2014.

[14] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *IEEE INFOCOM*, 2012, pp. 963–971.

[15] B. Martini, M. Gharbaoui, and P. Castoldi, "Cross-functional resource orchestration in optical telco clouds," in *17th Int. Conf. on Transparent Optical Networks (ICTON)*, July 2015, pp. 1–5.

[16] R. Vilalta, A. Mayoral, R. Munoz, R. Casellas, and R. Martinez, "Multitenant transport networks with SDN/NFV," *J. Lightwave Technol.*, vol. 34, no. 6, pp. 1509–1515, Mar. 2016.