The Need of a Transport API in 5G for Global Orchestration of Cloud and Networks through a Virtualised Infrastructure Manager and Planner (Invited)

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

***Abstract*—The new 5G paradigm seeks for a scalable architecture able to efficiently manage the increasing volume of traffic generated by smart devices to be processed in a distributed cloud infrastructure. To this end, a coordinated management of the network and the cloud resources forming an end-to-end system, is of great importance. Software Defined Networking (SDN) and Network Function Virtualization (NFV) architectures are the key enablers to integrate both network and cloud resources, enabling cross-optimization in both sides. This optimization requires efficient resource allocation algorithms which take into account both computing and network resources.**

In this paper, we propose an end-to-end orchestration architecture for distributed cloud and network resources aligned with the ETSI Management and Orchestration (MANO) 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 (VMG) resource allocation is included to validate the VIMaP architecture and exploit its functionalities. Moreover, the Control Orchestration Protocol (COP) 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.

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) or cloud resources (e.g, distributed nodes with cloud capabilities, edge /core data centers - DC). Thus, one of the main challenges will be to provide multiple, highly flexible, end-to-end dedicated network and cloud infrastructure slices over the same physical infrastructure in order to deliver application-specific requirements.

Manuscript received June X, 2016.

Arturo Mayoral, Raul Muñoz, Ricard Vilalta, Ramon Casellas, Ricardo Martínez are with CTTC, Castelldefels, Spain (e-mail: arturo.mayoral@cttc.es).

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

Software Defined Networking (SDN) architecture is the key enabler to integrate both network and cloud resources. SDN allows centralized programmability of the network by decoupling the data plane from the control plane through standard protocols (e.g., OpenFlow). The control entity (SDN controller) is responsible for providing an abstraction of the network forwarding technologies (e.g., packet/flow or circuit switching) through an Application Programming Interface (API). This abstraction enables network virtualization, that is, to slice the physical infrastructure and create multiple co-existing virtual tenant networks (VTN) independent of the underlying transport technology and network protocols. Ideally, the SDN architecture is based on a single control domain comprising multiple network nodes featuring diverse technologies provided by different vendors that are controlled through standard interfaces. However, it is not realistic in the short term in optical networks since they are fragmented into multiple vendor domains. The transport equipment does not interoperate at the data plane level (only at the grey 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 PCE) because of the need of configuring vendor-proprietary parameters (e.g., FEC), generating vendor islands.

SDN orchestration has been demonstrated as a feasible and scalable solution for multi-domain, multi-technology network scenarios to provide end-to-end (E2E) network services in [2] and [3]. A multi-domain 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). The conceived multi-domain SDN orchestrator architecture is based on the Application-based Network Operations (ABNO) [4] proposed in the Internet Engineering Task Force (IETF). Typically, the northbound interface (NBI) of a domain controller is technology and vendor dependent, so the multi-domain SDN network orchestrator has to implement different plugins for each of the domain controller's NBI [5]. The STRAUSS project (http://www.ict-strauss.eu) has defined the first unified Transport API named Control Orchestration Protocol (COP), that abstracts the particular control plane technology of a given transport domain.

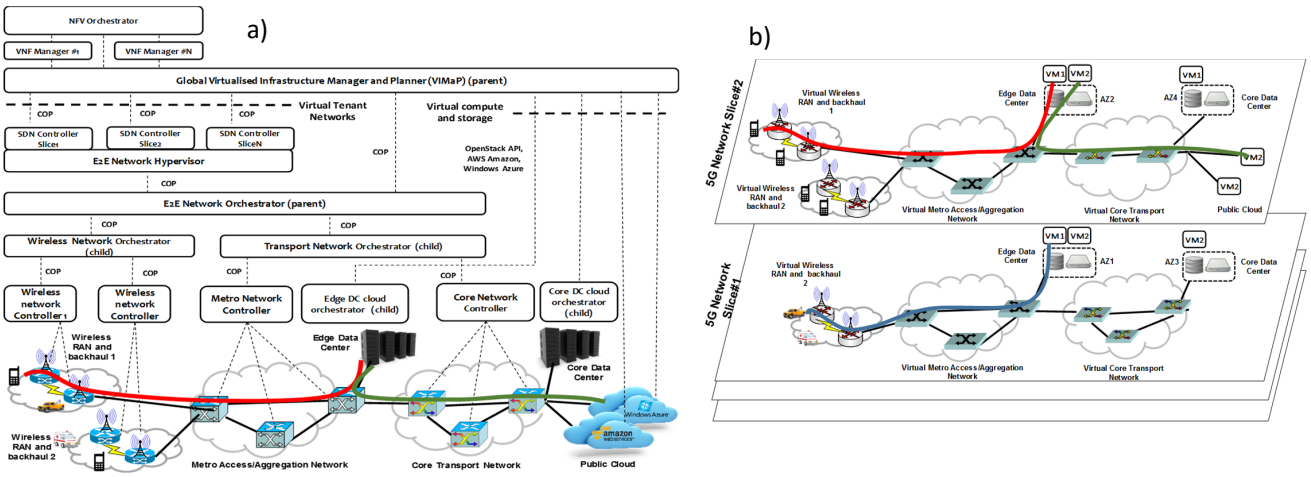

Fig. 1: a) 5G network architecture for dynamic provisioning of virtual compute, storage and network across distributed cloud infrastructures and heterogeneous networks; b) 5G network slices supporting different cloud and network requirements

COP provides a research-oriented multi-layer approach using YANG/RESTconf. The latest OIF/ONF Transport SDN API [6] is in line with COP's objectives. The COP definition is open for discussion and can be downloaded and contributed at https://github.com/ict-strauss/COP. The COP also enables the integration of heterogeneous radio access networks (5G, mmWave, LTE/LTE-A, Wi-Fi, etc) with transport networks as well as the orchestration of cloud resources and transport networks for DC interconnection.

In this paper, we propose an end-to-end orchestration architecture for distributed cloud and network resources aligned with the ETSI NFV Management and Orchestration (MANO) architecture (see Figure 1.a) [7]. The NFV architecture proposes a Virtualised Infrastructure Manager, which is responsible for managing the NFV infrastructure resources, such as compute, storage and networking. In our proposed architecture, we extend the Virtual Infrastructure Manager and Planner (VIMaP) component to enable dynamic resource allocation for interconnected virtual instances in distributed cloud locations. This component includes a specific module for resource optimization and planning, which provides an online platform to run resource allocation algorithms for the arriving requests.

The contribution of this paper in this topic is two-fold. On one hand, we have extended the work in [8], where the proposed SDN hierarchical architecture using the COP as an unified interface has been complemented by the introduction of theVIMaP). On the other hand, the second contribution consists on modeling the resource allocation problem of dynamically provisioning of Infrastructure-as-a-Service (IaaS), which in this paper we refer as Virtual Machine Graphs (VMGs) provisioning problem. In this paper, we propose an heuristic baseline solution based on greedy approach for the selection of DCs, combined with a First Fit (FF) for the virtual machine allocation. Based on the virtual machine allocation the Constrained Shortest Path First (CSPF) algorithm is employed to guarantee enough bandwidth for each connection between virtual instances allocated in different DCs.

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 multi-domain, multi-layer and multi-vendor 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. In brief, COP is composed of three main base functions:

1) Topology providing 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, providing an interface to request and return path objects which contain the information about the route between two endpoints;
3) Call, based on the concept of Call/Connection separation, and providing a common provisioning model which defines an end-to-end connectivity provisioning service.

The proposed COP provides a common NBI API so that all domain controllers can be orchestrated using a single common protocol.

One benefit of this architecture resides on the ability to perform unified control and management tasks (e.g., end-to-end 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 also desired to consider a hierarchical orchestration approach with different levels of hierarchy (parent/child architecture). Each successively higher level has the potential for greater abstraction and broader scope (e.g., we may considered 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 NBI of the child SDN orchestrator and as 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 multi-layer (packet/optical) and multi-domain transport provisioning across heterogeneous control domains (SDN/OF and GMPLS/AS-PCE) employing dynamic domain abstraction based on virtual node aggregation in [9].

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 multi-domain network hypervisor architecture has been proposed and assessed in [10].

Virtualization of compute, storage and networking resources in DCs is provided by private clouds through distributed cloud orchestrators (children) that may be deployed with different software distributions (e.g. OpenStack, OpenNebula), or by public cloud. Each cloud orchestrator enables 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 compute, storage and network 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 multi-layer VTN (Fig.1.b). A key enabler of such an integration is the COP, which is used as 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 [11] and [12].

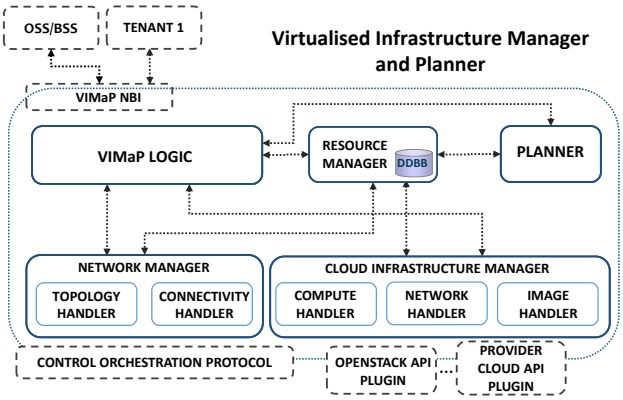

Fig. 2: VIMaP internal architecture, building blocks.

III. Virtual Infrastructure Manager and Planner

In this section we present the VIMaP architecture including the description of its main building blocks, which are showed in Figure 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 Virtual Machine Graph (VMG). The VIMaP architecture includes a Planner component dedicated to perform resource planning optimization. Different policies may be applied resource optimization including QoS guarantee, minimize network resources consumption or energy efficiency among others. The VIMaP architecture allows the VIMaP LOGIC component to select the preferred algorithm depending on the tenant preferences. It receives the resource allocation requests from the VIMaP logic and it obtains all the substrate infrastructure information from the Resource Manager component which maintain 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 the responsible of orchestrate the workflows among the different architectural components in order to provision the cloud and network resources for an upcoming request. It is the responsible for performing context-aware orchestration, exposing to each tenant only those resources allocated to the tenant by means of virtual representation. It includes a northbound interface (NBI) which exposes the custom set of VIMaP programmable resources to each tenant.

The Resource Manager is responsible for storing and maintaining 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 two-fold: first it provides the southbound interface towards network infrastructure controllers including the necessary application programmable interfaces (API) or protocols implementations. As we have presented before, the COP is the protocol chosen to unify the network orchestration interface towards different SDN controllers. Secondly, 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.

Cloud Infrastructure Manager is responsible for distributed cloud orchestration. Differently to the Network Manager, it is responsible of 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 of 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 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 Virtual Machine Graph (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 $c_a(h^x), a \in A, h^x \in H^X, A \in \{CPU, MEM, STO\}$ (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 constrains 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 of the problem which consist in: a) finding a VM allocation among the substrate hosting nodes and, b) find an 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 powerset 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., that satisfies 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 hosting and link mapping as:

- **Hosting mapping function**:
$$M^H : (H') \mapsto (H^{S'})$$
which satisfies:
$$\forall h' \in H', \forall h^{s'} \in H^{S'}, \quad \sum_{\forall h^v \in h'} c_a(h^v) \leq c_a(h^{s'}) \quad (1)$$
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, γ to weight up the CPU, Memory and Storage capacities respectively:
$$weight(h^x) = \alpha c_{CPU}(h^x) + \beta c_{MEM}(h^x) + \gamma c_{STO}(h^x) \quad (2)$$
- **Link mapping function**:
$$M^L : (L') \mapsto (P^{S'})$$
which satisfies:
$$\forall l' \in L', \forall p^{s'} \in P^{S'}, \quad bw(l') \leq BW(p^{s'}) \quad (3)$$
$$where, BW(p^s) = \min_{\forall l^s \in p^{s'}} bw(l^s)$$

C. Baseline VMG Embedding algorithm

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

We asses the problem in two steps:

- Step 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 First Fit approach (see Algorithm 1).
- Step 2: Based on the selected $H^{s\prime} \subseteq H^S$ 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 GreedyFFHostMapping(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.
  $H^{S\prime} \leftarrow \emptyset$
  $H' \leftarrow \emptyset$
  $H^{v\prime} \leftarrow H^V$
  while $\sum_{\forall h^{s\prime} \in H^{S\prime}}(c_a(h^{s\prime})) < \sum_{\forall h^v \in H^V}(c_a(h^v))$, $\forall a \in A$ **do**
    $h^s \leftarrow H^S.pop()$
    $currentC_a \leftarrow c_a(h^s), \forall a \in A$
    $current_s \leftarrow \emptyset$
    for v **in** $H^{v\prime}$ **do**
      if oneOf $currentC_a < c_a(v), \forall a \in A$ **then**
        $H^{v\prime} \leftarrow H^{v\prime} - current_s$
        break
      else
        $current_s \leftarrow current_s \cup \{v\}$
        $currentC_a \leftarrow currentC_a - c_a(v), \forall a \in A$
      end if
    end for
    $H' \leftarrow H' \cup current_s$
    $H^{s\prime} \leftarrow H^{s\prime} \cup h^s$
  end while
  return $M^H : H' \mapsto H^{S\prime}$

Algorithm 1 first computes the Greedy and the First Fit (FF) host mapping procedure to find the minimum cluster with enough capacity to allocate virtual hosts within the VMG request. Firstly, it sorts the substrate host set in decreasing order by weigh 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 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 constrain parameter. If there is a feasible path for each $l' \in L'$, the mapping solution is returned : $(H', L') \mapsto (H^{S'}, P')$.

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

D. VMG allocation results

Parameter	Values
H^S CPU values	[100, 200, 400]
H^S Memory values	[200, 400, 800]
H^S Storage values	[10000, 20000, 40000]
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

TABLE I: Experiments parameter configuration

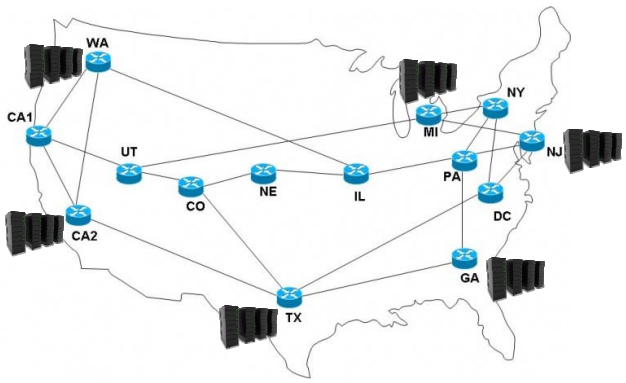

Fig. 3: NSF Network of 14 nodes with 6 DC

In this section we present the evaluation of the proposed heuristic baseline solution. The substrate infrastructure scenario employed for the experiments is an extended version of the NSFNET of 14 nodes and 42 unidirectional links and 6 DCs (Figure 3). For simplicity, the DCs are co-located within the same network node locations and the connectivity between DC's and its corresponding network nodes is modelled to have infinite bandwidth. The substrate infrastructure is initially configured with a pre-defined 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 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 are randomly connected with probability 0.5, in total we will have $n(n-1)/4$ links in average. The capacities of the virtual hosts and the virtual links are also selected randomly following an uniform distribution along the values depicted in Table I.

The VMG requests arrives to the VIMaP following a Poisson process on which the arrival rate is varying. The holding time of the VMG requests in the system follows an exponential distribution with 10 time windows on average. We run all the simulations for 10000 requests for each instance of the simulation.

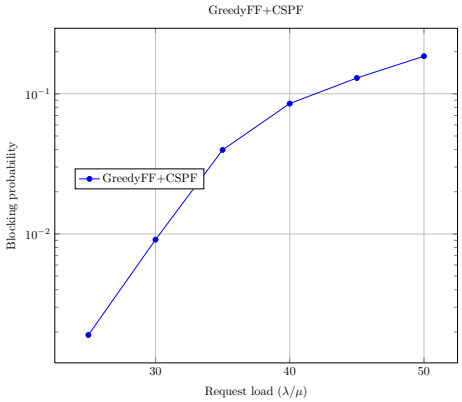

Fig. 4: VMG request blocking results for the GreedyFF+CSPF baseline heuristic algorithm

The results of the simulation for different loads can be seen in Figure 4. The algorithm's behaviour is as it was expected allowing the effective allocation of cloud and network resources based on the current situation of the substrate infrastructure. 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 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 controlled using OpenStack (Havana release), which has been deployed into servers with 2 x Intel Xeon E5-2420 and 32GB RAM each. An Openstack controller node and four compute nodes have been setup 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 2 ROADMs and 2 OXCs. The multi-domain SDN orchestrator (MSO) and VIMaP entities have been mostly implemented in Python with the exception of the Multi-layer PCE which has been implemented in C++ [13].

The COP has been employed as a Transport API for the orchestration of: two SDN OpenDaylight Helium controllers responsible of 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 controllers NBI. Figure 5.a shows a multi-domain 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 firstly 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)). Afterwards 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 showed in Figure 6 validates the use of the COP. Firstly, we can observe the request for Virtual Machine (VM) creation from VIMAP towards the Cloud Controller (which is running on the same server). The creation time for a single VM is of 15 seconds, which include the necessary time to boot up the VM. Secondly, in Figure 7 we can observe the Call request (Call Id: 1) from the VIMaP towards the multi-domain 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 towards the AS-PCE for the optical domain (Call Id: 00002, 00005), the SDN Controller 1 (Call Id: 00001, 00006) and the SDN Controller 2 (Call Id: 00003, 00004). The multi-domain call service set-up delay is of 2.52 seconds.

VI. Conclusion

The definition of a Transport API that abstracts a set of control plane functions used by an SDN Controller, allowing the SDN orchestrator to uniformly interact with heterogeneous control domains will pave the way towards the required transport network interoperability as well as the 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.

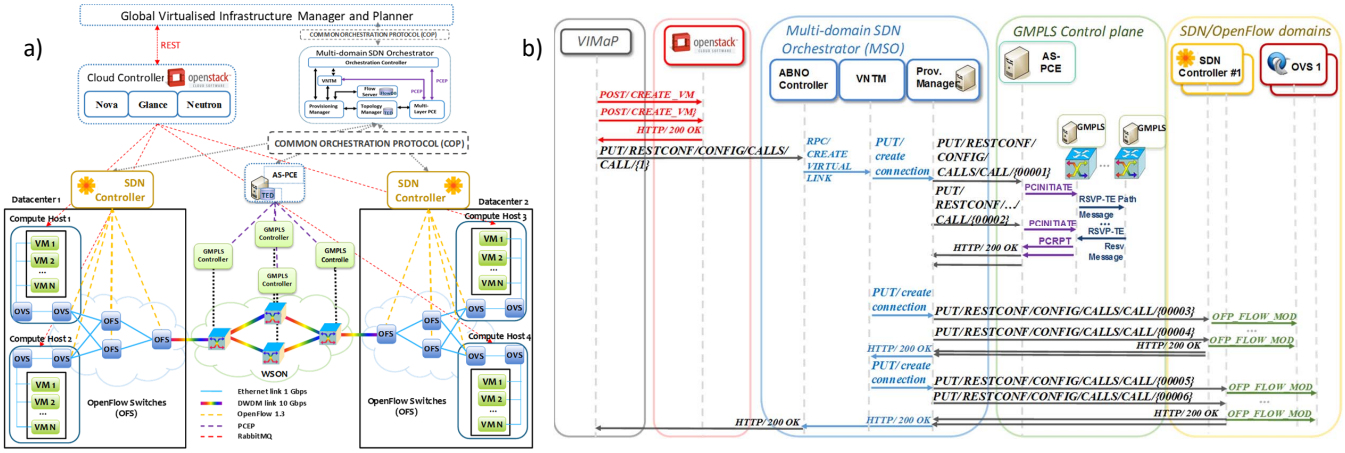

Fig. 5: a) Experimental scenario for DC interconnection; b) Integrated IT/SDN orchestration workflow

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

Fig. 7: Example of JSON COP Call object

ACKNOWLEDGMENT

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

REFERENCES

[1] N. Alliance, "5g white paper," *Next Generation Mobile Networks, White paper*, 2015.

[2] R. Muñoz, R. Vilalta, R. Casellas, R. Martínez, 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," *Journal of Lightwave Technology*, vol. 33, no. 8, 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," *Journal of Lightwave Technology*, vol. 33, no. 3, pp. 609–617, 2015.

[4] D. King and A. Farrel, "A pce-based architecture for application-based network operations," 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 Conference*, pp. Th2A–41, Optical Society of America, 2015.

[6] OIF-ONF, "White paper: Global transport sdn prototype demonstration," 2014.

[7] ETSI Group Specification, "Network function virtualization (nfv): Architectural framework," *ETSI GS NFV 002 v.1.1.1,*, 2013.

[8] 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 Conference*, Optical Society of America, 2016.

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

[10] 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]," *Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. B55–B61, 2015.

[11] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, and R. Martinez, "Experimental seamless virtual machine migration using a sdn it and network orchestrator," OFC, 2015.

[12] 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 *Proc. European Conf. on Optical Communication (ECOC)*, 2014.

[13] R. Vilalta, A. Mayoral, R. Munoz, R. Casellas, and R. Martinez, "Multitenant Transport Networks with SDN/NFV," *IEEE/OSA J. of Lightwave Technology*, vol. 34, March 2016.