End-to-end Service Orchestration From Access to Backbone

J. M. Gran Josa, V. Lopez
Telefonica I+D
Ronda de la Comunicación, s/n
Madrid, Spain

F. Slyne, M. Ruffini
Trinity College Dublin
Dublin, Ireland

R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R.Martínez
CTTC
Av. Carl Friedrich Gauss, 7
Castelldefels, Spain

***Abstract*—Due to the importance for network operators of dynamically control networks (in contrast to what has been deployed until now), this paper proposes a technical solution based on an SDN architecture. With the introduction of an end-to-end (E2E) SDN architecture, it is possible to support an E2E service from access to backbone networks. This paper firstly presents an optimum end-to-end network architecture, which covers both network domains. A transport API is introduced (Control Orchestration Protocol protocol) and it is used for joint network orchestration between network controllers, in order to provide these E2E services. Finally, this paper presents a multi-partner demonstration, which involves two different labs, located in Telefonica and Trinity College of Dublin facilities. This demonstration is used to validate this work.**

Keywords— SDN; PCE; ABNO; Network Orchestrator; Network Controller; Transport API; COP.

I. INTRODUCTION

Network operators have deployed their networks to cope with static services, where the traffic sources were clearly defined and they were not moving along the years. These types of services are changing and there is an important trend to deploy data centres to enable the deployment of application or virtual network functions (using network function virtualization – NFV). These new services and the expected continuous growth in Internet traffic (mainly driven by video) will imply a huge impact on the end-to-end (E2E) network. The network costs of current architectures depend significantly on traffic growth; the higher the traffic the higher the network costs. Consequently, any cost increase will impact on the ISP's margins. The network architectures of the last 20 years were never designed to cope with these new types of service demands. Therefore, new architectural solutions are needed to deliver the huge expected increase in traffic in a cost-effective way, and ensure low cost broadband Internet access in Europe. The DISCUS project presents in [1] a metro-core node proposal to enable a most optimum E2E network architecture.

In order to have a dynamic node solution, it is mandatory to have a control plane that enables the configuration of the resources on demand. Software Defined Networking (SDN) proposes to decouple the control plane from the data plane to allow this dynamic operation. The utilization of SDN for access networks is motivated from operators [2], who are willing to have dynamic access networks to deploy new services. Recent works [3,4] have presented SDN demonstration in access networks.

Moreover, the utilization of GMPLS in core optical networks has allowed having a SDN ready solution for the backbone network. The appearance of Path Computation Element (PCE) and Application Based Network Operation (ABNO) architecture allows having a SDN solution for fix and flexi grid networks. There are recent works [5, 6] demonstrating the dynamic provisioning of services in backbone networks, considering multi-layer and multi-technology environments.

This paper defines and demonstrates an architecture to support E2E service from access to backbone. The remaining of the article is structured as follows. Section 2 introduces the DISCUS metro-core node architecture. Section 3 defines the SDN architecture to allow end-to-end services. This work uses the Control Orchestration Protocol, which is presented in Section 4. Section 5 introduces the use case that motivate the work. Section 6 presents the experimental validation and the results. Finally, Section 7 concludes this article.

II. METRO/CORE NODE ARCHITECTURE

The DISCUS metro/core nodes are core edge nodes in a similar architectural position in the network to what are often called metro-core nodes in classic telecommunication architectures. The DISCUS Metro/Core node are the only nodes in the network covered by a single optical island with traffic processing functions. The architecture of these nodes is flexible enough so that different (IP, Ethernet and Optical) layers can evolve and if necessary displace other layers minimising cost and energy consumption. The node architecture consists of an optical switching layer, an Ethernet layer and an IP layer. The optical switch provides flexible interconnect between theses layers and the optical channels from the access and core networks. The large port-count optical switch allows maximum flexibility as any incoming fibre can be terminated, after demultiplexing, at any OLT, or can be re-amplified and sent back to another ONU or regenerated and sent over the optical core network. Since every access PON can carry a large number of wavelengths, potentially 80 or more in the medium to long term, the optical switch must be highly scalable, while offering a very low optical loss (less than 2-3dB). The Optical Switch should have large switch matrices and a potentially be 3-stage switches capable of scaling to over 12000 ports.

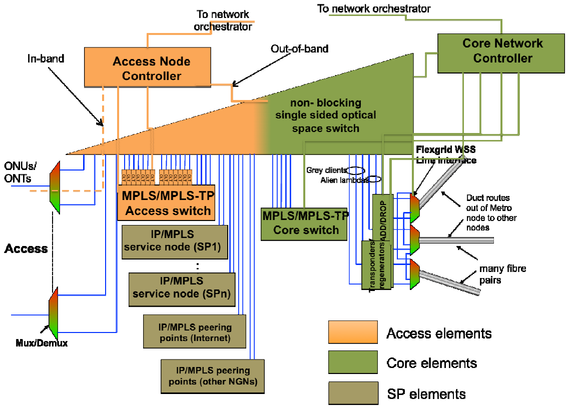

To network orchestrator
To network orchestrator
In-band
Access Node Controller
Out-of-band
Core Network Controller
non- blocking single sided optical space switch
ONUs/ ONTs
Access
Mux/Demux
MPLS/MPLS-TP Access switch
IP/MPLS service node (SP1)
IP/MPLS service node (SPn)
IP/MPLS peering points (Internet)
IP/MPLS peering points (other NGNs)
MPLS/MPLS-TP Core switch
Grey clients
Alien lambdas
Transponders regenerators
ADD/DROP
Flexgrid WSS Line interface
Duct routes out of Metro node to other nodes
many fibre pairs
Access elements
Core elements
SP elements

Fig. 1. DISCUS metro/core node architecture

An access network controller is associated to every metro/core node, where it controls the optical switch, access switch, and OLTs/ONUs. The access controller sends abstracted topological information about the resources available within its domain. Where access protection is required, the controller handles incoming failure messages from the OLT to operate fast protection. Moreover, the access controller receives provisioning requests from the orchestrator and reports the service setup status. Finally, the access controller should be able to carry out path computation, because the network orchestrator can request a path computation.

To do so, the physical domain information have to be obtained from the network elements and mapped in the abstracted view. Similarly, the provisioning of abstracted services is map in real configurations. Therefore, the access controller maintains information on bandwidth availability within the access switch and each OLT. Besides, it configures the network elements (access switch, optical switch, OLT, ONU/ONT). depending on its specific requirements.

III. SDN Architecture to Allow E2E services

The SDN control plane architecture is shown in Fig. 2, and it is based on a hierarchical structure of controllers. This architecture is aligned with Open Network Foundation (ONF) work. Three main logical components can be identified for the SDN control plane. The access network controller, in charge of controlling the access network elements; the backbone network controller, in charge of controlling the elements carrying out backbone transmission and the network orchestrator, in charge of taking requests from application plane and translating them into high-level commands for the access and backbone network controllers.

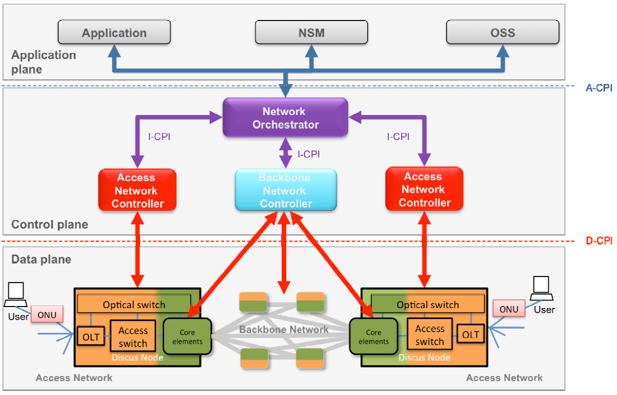


Fig. 2. Proposed SDN Architecture

In terms of interfaces, the overall architecture follows the ONF architecture using three main interfaces: A-CPI, I-CPI and D-CPI. The A-CPI interface describes the interaction between the control plane and the application plane. Thus is the interaction between the service provider and the network orchestrator. The I-CPI interface operates between the network orchestrator and the access and core network controllers. Lastly, the D-CPI interface operates between the access and backbone controllers and the physical devices. There are different protocols to cope with the functionalities in each interface. The demonstration in this work has selected a set of protocols to fulfil the requirements in each domain aligned with the industry trends.

A. *Network Orchestrator*

The network orchestrator is defined as a parent controller or a centralized "controller of controllers", which handles the automation of end-to-end connectivity provisioning, working at a higher, abstracted level and covering inter-domain aspects between the access and the metro/core network. The network orchestrator interfaces with the controllers to get topological information about the resources in each controller's domain. Each controller may have different interfaces, which requires the orchestrator to have a method to support multiple technologies or interfaces.

When an application, such as Network Management System (NMS) or Operation Support System (OSS) requests a service, the network orchestrator must deal with the end-to-end path computation and service request. This process can be done by the orchestrator or delegated to the access and backbone controllers. Once the services are set-up, the network orchestrator is in charge of update its status and notify to the application plane.

B. *Access Network Controller*

The DISCUS SDN Access Network Controller (NC) show in Fig. 3 translates requests from the Network Orchestrator (NO) into instructions for the physical devices, such as access switch, optical switch and OLT. The NC consists of JSON (REST/API), Application module, Database, and RYU controller. The JSON (REST/API) module is an interface that translates a JSON request, which is received from the NO via I-CPI which is then sent to the application module.

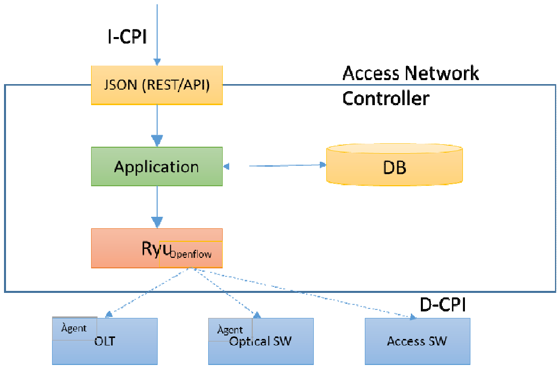


Fig. 3. DISCUS Metro Access Network Controller

The application module processes the incoming request, based on the state information present in the database. This module implements functionalities such as path calculation, path recovery, wavelength selection, bandwidth assignment, PseudoWire (PW) assignment and Link State Protocol (LSP) assignment. Based on the request and the network state, it determines whether the request can be satisfied or should be declined. The application module triggers the appropriate OpenFlow commands using the RYU OpenFlow controller. The NC sends an acknowledgment message back to the NO via I-CPI. The NC maintains openflow rules and meters in the access switch. A two-stage hierarchy of meters implements Peak Information Rate (PIR) and Committed Information Rate (CIR) Quality of Service characteristics for each flow. PIR is implemented by discarding packets that exceed the first meters bitrate. CIR is subsequently implemented by marking non-dropped packets as low priority (prec_level=0) that exceed the CIR bitrate defined by a second meter. The database module stores all information from the MC node on routing, wavelengths, capacity, MPLS labels, and detail of flows and meters being used. Infrastructure information is held in database tables and relates to topology and the definition of ports, paths and host configuration. The paths table defines the route between sources and destinations. The switch ports table defines the port connectivity of each switch. In the current implementation, this information is statically defined but this information could be gathered through discovery protocols, in more complex implementations. The Host IP table defines the IP address for each numbered host. host_id indicates the ID number of the host (it could be an SP server, ONU, or OLT); ip shows the IP address of the host; PRI_OLT_IP shows an IP address of a primary OLT (if the device is an ONU). The Access Network controller uses openflow to manage the various network components in the access network, such as the optical switch, access switch and PON components. We have devised an Openflow Agent that provides a mediation interface between the upper Openflow Control plane and the lower non-native openflow network devices such as the FPGA-based LR-PON OLTs and ONU. This interprets the openflow commands coming from the Access Network controller and communicates various changes to the PON network via a UART control link in the FPGA hardware. The OLT and connected ONUs can be controlled via the access network controller like any other network components.

C. Backbone Network Controller

The backbone controller is in charge of receiving commands from the network orchestrator and transforming them in the D-CPI for the metro/core network. Similarly, it exports the topology to the network orchestrator, so it can have a view of the resources in the backbone network. The network orchestrator can request a path computation to the backbone controller, so it must support path computation within its domain.

A backbone controller is used as an entity, which is in charge of the specifics of the underlying backbone technologies. The technologies under the backbone controller are Optical Transport Network (OTN), Wavelength Switched Optical Network (WSON), Spectrum Switched Optical Network (SSON) networks, which are based on the GMPLS distributed control plane. If the GMPLS is enabled, the best interface with the nodes is Path Computation Element Protocol (PCEP), as demonstrated in [5].

IV. Control Orchestration Protocol

The Network Orchestrator is in charge of the control (e.g., E2E transport service provisioning), at a higher, abstracted level, of end-to-end resources across multiple domains with heterogeneous multi-layer multi-vendor transport network technologies regardless of the specific control plane technology employed in each domain. Typically, domain SDN controllers have a northbound interface (NBI) which is technology and vendor dependent, so the Network Orchestrator has to implement different plugins for each of the domain SDN controller's NBI [7]. During the latest OIF/ONF Transport SDN demonstration [8], vendors tested prototypes of SDN controllers NBI for Service Request and Topology functions in development by the OIF. The framework of the demo was application-based bandwidth-on-demand between data center sites.

The Control Orchestration Protocol (COP) [9] is the first defined Transport API, which abstracts the particular control plane technology of a given transport domain. COP provides a research-oriented multi-layer approach using YANG and RESTconf. It can be both seen as a solution for the I-CPI, between the Network Orchestrator and domain SDN controllers, as well as a detailed A-CPI for an application (i.e., NMS, OSS) running on top of the Network Orchestrator. Moreover, 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 resources for DC interconnection.

In brief, the COP is composed of three main base functions:

1) *Topology service*, which provides topological information about the network, including a common and homogeneous definition of the network topologies included in the Traffic Engineering Databases (TED) of the different control instances;

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

The COP definition is open for discussion and can be downloaded and contributed at https://github.com/ict-strauss/COP.

The **Topology Service** provides abstracted topological information about the network. It must include a common and homogeneous definition of the network topologies included in the TED of the different control instances. The COP Topology provides the interface for the exchange of detailed network topology between SDN controllers. An abstract Topology object may consist of a set of nodes and edges, which form a tree structure. A Node must contain a list of ports/endpoints and their associated switching capabilities (e.g., packet/lambda switch capable). An Edge object is defined as the connection link between two Endpoints, which includes some traffic metrics and characteristics. Due to the need of conforming to a common model among different transport network technologies, the definition of the three main objects described (Node, Edge, Endpoint) must be extensible, able to include TE extensions to describe different switching capabilities (i.e., time-slots, packets, wavelengths, frequency slots).

The **Call service** is defined in the scope of COP as the provisioning interface of each domain SDN controller. The Call is defined as the API's basic data-container and it completely solves the provisioning of an End-to-End connectivity service. A Call object describes the type of service that is requested or served by it (e.g., DWDM, Ethernet, MPLS). It also contains the endpoints between whom the service is provided. The Call object also includes the list effective connections made into the data plane, to support the service call. Moreover, a Call also includes a set of traffic parameters that need to be supported, such as Quality of Service, or allocated bandwidth. Finally, a Call shall include the Match parameter, which refers to the traffic description to which the Call refers. A Connection object is used for a single network domain scope. It should include the path object (i.e., the route) across the network topology the data traverses, which may be fully described or abstract depending on the orchestration/control schemes used. Each connection must be associated with a single control plane entity (e.g. a SDN controller) responsible for the configuration of the data path. Finally, the Call also introduces the necessary TE parameters (e.g., bandwidth) that the service requests.

Both the Topology and Call services can be extended in order to not only provide abstracted information, but to provide low level technological dependant details, such as optical spectrum grid or CIR/PIR.

Finally, the **Path Computation** service should provide an interface to request and return Path objects, which contain the information about the route between two Endpoints. Path computation is highly related to the previous group of resources. In the service Call, the Connection object has been designed to contain information about the traversed Path. Furthermore, each component in the Path object is represented as an Endpoint with TE information associated to it.

V. USE CASE FOR DEMONSTRATION

The Use Case shown in Fig. 4 exemplifies how capacity constraints in the Core and Access Metro network of a Telecommunications network may be overcome both in terms of restoration of QoS but also opportunistically for the dynamic temporary provision of high bandwidth services. In the Metro Access network, we implement a DWA mechanism that can facilitate allocation of capacity (an entire wavelength or a portion of a TDM-PON channel) to specific PON users on a dynamic and temporary basis, with applications for on-demand multimedia and big data transfers. Alternative scenarios envisaged include the reapportionment of wavelength bandwidth throughout the PON network by time of day, week or month. In the cases of temporary and dynamic migration of wavelength it is important that the wavelength switching time is minimized to avoid impacting users of other network services. In Step 1, an end user (customer) elects through a portal frontend to the ABNO controller to transmit a fixed bandwidth transport (100Mbps) between two end points aEnd (10.0.50.3) and zEnd (10.0.50.4) typically for the sending and receipt of Video streaming. In Step 2, the ABNO orchestrates the provision of the path, according to its knowledge of the full End to End topology covering the Core and Metro Access Networks. The Idealist core backbone NC and the Metro Access (DISCUS) SDN NC, are each instructed to provision an explicit path in steps 3 and 4 respectively. For the Metro Access portion of the network (the path from 10.0.50.2 to 10.0.50.4), the DISCUS SDN NC is instructed to provision the path according to the route 10.0.50.2 to 10.0.50.1 to 10.0.50.4. The DISCUS SDN Controller provisions the path through the Metro Access (openflow) switch (step 5), and the PON (primary/secondary OLT and ONU – step 6). Using a custom implemented PLOAM message, the primary OLT requests the ONU tune to a wavelength provisioned out of the secondary OLT (step 7). In step 8, the DISCUS SDN NC, acknowledges that it has completed the provisioning of the path through the Metro Access portion of the network. In step 9, the Video transmission is triggered to start.

VI. EXPERIMENTAL VALIDATION

A. Test-bed

There are two different labs set-up to demonstrate the scenario of this work one in Telefonica premises and another in Trinity College of Dublin. Fig. 4 shows a schema of the lab set-up for this experiment. To communicate them, there is a VPN created, so the data plane connection has a low bandwidth.

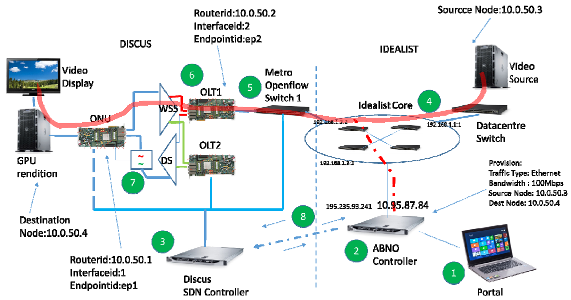


Fig. 4. Experimental laboratory set-up

The network orchestrator is located in Telefonica labs and is based on netphony ABNO implementation[1]. The north and south-bound interface of the orchestrator is implemented using the STRAUSS COP.

The backbone controller uses the netphony ABNO, which in addition with netphony PCE controls a GMPLS emulated control plane. The GMPLS nodes uses the protocol suite developed by Telefonica I+D and is released in github[2]. This setup is built with 30 virtual machines, which run in a Linux server distribution. Each emulated node implements a GMPLS stack (including RSVP, OSPFv2 and PCEP) with the extensions to support flexgrid developed in IDEALIST project.

The LR-PON Protocol is implemented over three Xilinx VC709 FPGA boards acting as primary and secondary OLTs and ONU. The LR-PON protocol is a partial implementation of the XGPON standard, the major differences being that the LR-PON protocol must work over a longer feeder fiber (125Km in LRPON as opposed to 20Km in XGPON) and across a higher split ratio (512 versus 64). The PON backplane connection to the core network contains a 10G Ethernet physical layer and Media Access Control Layer, allowing it to be plugged into any 10G capable network element. In this experiment the PON backplane is connected to a 48-port 10G Openflow Access/Metro switch. A Microblaze soft processor, which is collocated on the Virtex FPGA board, provides a (North Bound) UART management interface to the PON OLT and ONU hardware. Through this interface most PON functionality can be controlled such as resetting the hardware, viewing hardware status, simulating hardware failure, loading bandwidth map and setting XGEM mappings. Dynamic Wavelength Assignment is supported through the addition of laser and filter control to the LR-PON protocol hardware and control mechanisms. The tunable laser is controlled across an i2c bus to the Skylane 10G SFP+ tunable lasers and the tunable filter is controlled through a UART. To implement DWA in the physical layer, we employ a splitter and filter in the downstream, and a WSS (Optimum 9x1/1x9 50Ghz Wavelength Selective Switch) in the upstream direction, statically patched to ITU channels 32.5 and 31 of the primary and secondary OLT respectively. In order to select the OLT and ONU transmission wavelengths, the OLT provides a North Bound interface. Through this Interface, the control plane can tune an OLT's transceivers to a given wavelength. Since the ONU is remote from the control plane, tuning of an ONU's laser and filter can be performed also through this interface by the invocation of custom PLOAM message within the LR-PON protocol. The wavelength of the OLT and ONU may be selected by writing to control registers in the OLT. Each individual OLT laser wavelength can be set by writing the ITU channel number to its local register. To select the wavelength of transmission for an ONU, the ITU channel number is set by writing the target ITU channel number to register of the OLT to which it is homed. The ONU Id must also be specified so as to distinguish an individual from multiple ONU's homed off a single OLT.

For the experimental LR-PON access network, the configuration is comprised of a Pronto 3780 switch with 48 10G interfaces, running release 2.4 (Openflow v1.4 compatible firmware). The Pronto switch is configured with multiple virtual bridges.

A Video Server (VLC) application is co-located with the ABNO controller interface in the Telefonica premises. This transmits a UDP based video stream across the Tunnel between the two testbeds, traversing the DISCUS PON and being received by the GPU workstation for display by the TV display. The latency between the two testbeds was measured over 100 measurements at between 45 and 48 ms. It was not possible to transmit the video stream through the Idealist network, as a physical DataPath was not available. .

B. *Workflow steps in the architecture*

This section explains step by step how this demonstration is performed (Fig. 4). First, the portal requests a new video service, which cannot be processed within the access area scope. This means that there is a request of the video platform to provisioning an end to end path between the client and the video server. Therefore, the STRAUSS ABNO receives a COP calls service set-up to establish the connection. The STRAUSS ABNO carries out a path computation, which crosses different networks, the backbone (IDEALIST) and access network (DISCUSS). Therefore, the STRAUSS ABNO sends a COP calls service set-up to each controller to configure the nodes in their domain. The IDEALIST PCE configures the GMPLS nodes, while the DISCUS controller configures the access elements. The workflow is explained in Fig. 5.

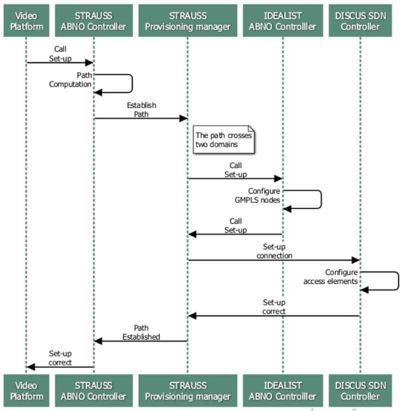


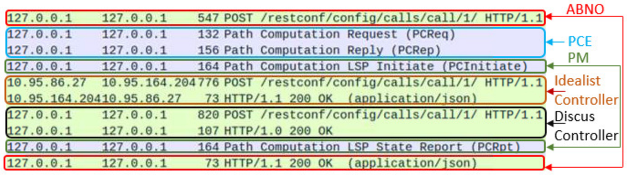
Fig. 5. Workflow steps

[1] https://github.com/telefonicaid/netphony-abno/wiki

[2] https://github.com/telefonicaid/netphony-network-protocols

C. Results

The previous workflow was demonstrated in the lab set-up. Fig. 6 shows the message exchange between the different elements.

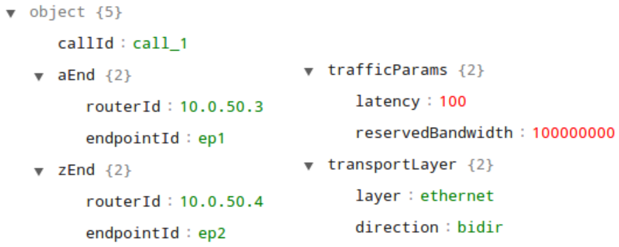


```
127.0.0.1     127.0.0.1     547 POST /restconf/config/calls/call/1/ HTTP/1.1
127.0.0.1     127.0.0.1     132 Path Computation Request (PCReq)
127.0.0.1     127.0.0.1     156 Path Computation Reply (PCRep)
127.0.0.1     127.0.0.1     164 Path Computation LSP Initiate (PCInitiate)
10.95.86.27   10.95.164.204776 POST /restconf/config/calls/call/1/ HTTP/1.1
10.95.164.20410.95.86.27     73 HTTP/1.1 200 OK  (application/json)
127.0.0.1     127.0.0.1     820 POST /restconf/config/calls/call/1/ HTTP/1.1
127.0.0.1     127.0.0.1     107 HTTP/1.0 200 OK
127.0.0.1     127.0.0.1     164 Path Computation LSP State Report (PCRpt)
127.0.0.1     127.0.0.1      73 HTTP/1.1 200 OK  (application/json)
```

Fig. 6. Whireshark capture

As it is shown in Fig. 6 , the ABNO receives an HTTP POST request with COP syntax. Fig. 7 shows the JSON object with the request parameters. The aEnd and zEnd routerIds identifies the client and the video server. The traffic parameters are set to request a 100Mbps connection and a 100 ms latency. This request is sent as an Ethernet service.

Once the STRAUSS ABNO receives the requests asks its PCE for a path computation between the two end points. To do so a PCReq-PCResp process is performed. Now, the PCE can calculate the path and response to ABNO with a PCResp, which contains the Explicit Route Object with the path. The ABNO controller with the ERO information call to Provisioning Manager (PM) via a PCInitiate message. The PM split the route in different domains and with a COP message call to each controller to create a path in each domain (IDEALIST and DISCUS). When the path is created each controller send respective http message with an OK status. With this information PM response to ABNO contoller with a PCEReport message and finally ABNO report to video platform with an HTTP response.



```
▼ object {5}
    callId : call_1
  ▼ aEnd {2}
      routerId : 10.0.50.3
      endpointId : ep1
  ▼ zEnd {2}
      routerId : 10.0.50.4
      endpointId : ep2
  ▼ trafficParams {2}
      latency : 100
      reservedBandwidth : 100000000
  ▼ transportLayer {2}
      layer : ethernet
      direction : bidir
```

Fig. 7. JSON object for a COP service-call set-up

Across 10 repetitions of the experiment, the total completion time of the workflow was measured at 275ms, Of this, 35 ms (with associated inter-testbed latency) related to the blocking element of the call to the DISCUS SDN Controller. The non-blocking elements of the DISCUS SDN proceed in parallel with the completion of the return calls by the ABNO controller.

VII. CONCLUSONS

The network costs of current architectures depend significantly on traffic growth. The DISCUS project presents in a metro-core node proposal to enable a most optimum end-to-end network architecture, but it requires the interaction with a backbone solution like the flexgrid technologies of IDEALIST.

This work defines and demonstrates an SDN architecture to support end-to-end service from access to backbone. The validation work has involved two laboratories and thanks to this work, we can claim that network operators can deploy SDN solutions that covers, not only access or backbone scenarios, but also end to end.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 318137 in DISCUS project, in. 317999 in IDEALIST research project and no 608528 by the Japanese Ministry of Internal Affairs and Communications (MIC) and National Institute of Information and Communications Technology (NICT) through the EU-Japan coordinated project STRAUSS.

REFERENCES

[1] Ruffini, M.; Wosinska, L.; Achouche, M.; Jiajia Chen; Doran, N.; Farjady, F.; Montalvo, J.; Ossieur, P.; O'Sullivan, B.; Parsons, N.; Pfeiffer, T.; Xing-Zhi Qiu; Raack, C.; Rohde, H.; Schiano, M.; Townsend, P.; Wessaly, R.; Xin Yin; Payne, D.B., "DISCUS: an end-to-end solution for ubiquitous broadband optical access," in *Communications Magazine, IEEE* , vol.52, no.2, pp.S24-S32, February 2014

[2] N. Bitar, SDN and Potential Applicability to Access Networks, Paper Th1G.4, OFC 2014.

[3] X. Li, et al., Joint Bandwidth Provisioning and Cache Management for Video Distribution in Software-Defined Passive Optical Networks, Paper Tu3F.5, OFC 2014.

[4] M. Ruffini, F. Slyne, C. Bluemm, N. Kitsuwan, S. McGettrick, Software Defined Networking for Next Generation Converged Metro-Access Networks, Optical Fiber Technology, Volume 26, Part A, December 2015, Pages 31-41.

[5] A. Aguado, V. López, J. Marhuenda, Ó. González de Dios and J. P. Fernández-Palacios: ABNO: a feasible SDN approach for multi-vendor IP and optical networks , in Journal of Optical Communications and Networking, February 2015, Vol. 7, Iss. 2, pp. A356–A362.

[6] 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 Achim Autenrieth: Transport Network Orchestration for End-to-End Multi-layer Provisioning Across Heterogeneous SDN/OpenFlow and GMPLS/PCE Control Domains, in Journal of Lightwave Technology, April 2015, Vol. 33, No. 8, pp. 1540 - 1548.

[7] V. López, O. González de Dios, L.M. Contreras, J. Foster, H. Silva, L. Blair, J. Marsella, T. Szyrkowiec, A. Autenrieth, C. Liou, A. Sasdasivarao, S. Syed, J. Sun, B. Rao, F. Zhang, and J.P. Fernández-Palacios, "Demonstration of SDN orchestration in optical multi-vendor scenarios", OFC 2015.

[8] "Global transport sdn prototype demonstration," OIF / ONF White Paper, 2014.

[9] R. Vilalta, V. López, A. Mayoral, N. Yoshikane, M. Ruffini, D. Siracusa, R. Martínez, T. Szyrkowiec, A. Autenrieth, S. Peng, R. Casellas, R. Nejabati, D. Simeonidou, X. Cao, T. Tsuritani, I. Morita, J. P. Fernández-Palacios, R. Muñoz, The Need for a Control Orchestration Protocol in Research Projects on Optical Networking , in Proceedings of European Conference on Networks and Communications (EuCnC 2015), 29-2 July 2015, Paris (France).