# Open Source Netphony Suite: Enabling Multi-layer Network Programmability

V. López, R. Jimenez, O. Gonzalez de Dios, L.M. Contreras, J.P. Fernandez Palacios
Telefónica gCTO
Ronda de la Comunicación s/n 28050 Madrid. Spain
Corresponding email: victor.lopezalvarez@telefonica.com

***Abstract*—Network operators must deal with multi-layer architectures in their production networks. Not so much time ago, network operators were delivering the IP traffic using ATM. ATM protocol was used to aggregate end-user traffic coming from DSL connections. This traffic was transported Synchronous Digital Hierarchy (SDH) as the standard transport technology for fiber-optic transmission systems in backbone networks. There are still networks operating this way. However, current deployments are based on an IP/MPLS layer, either on their own (using dark fiber) or supported by an optical switching network (WDM, OTN, etc.). The IP/MPLS switching and routing are the layers that take advantage of statistical multiplexing and maximize the utilization of the optical links created in the underlying layer.**

**This multi-layer architecture requires a network programmability layer that enables the control and management of the IP and the optical resources. The advent of distributed protocols to deploy the Internet, lead on many advantages. The main one is the resiliency capability of a network, where each individual node can make their own decisions. On the other hand, a central intelligence enables to have a complete network view to propose optimal solutions to improve the resource utilization. Our proposal of multi-layer network programmability contains a hybrid approach to lever on the advantages of both paradigms. In terms on network protocols, there are to main trends binary versus REST-based. Our view is that binary protocols improve the performance at the low levels on the network, while REST-based APIs enables a faster development and network interoperability.**

**The Open Source Netphony suite is composed by a GMPLS control plane to emulate the network elements control, a Path Computation Element with active and stateful capabilities, a Topology Module capable of importing and exporting TE information in different protocols as well as an Application-based Network Operations (ABNO) controller. This framework enables multi-layer programmability for IP and optical networks.**

***Keywords— Multi-layer networks, ABNO, PCE, SDN control, programability***

## I. INTRODUCTION

Network operators have deal with multi-layer architectures in their production networks since they were deployed. To provide DSL service to the customers, network operators were delivering the IP traffic on top of ATM. ATM protocol was used to aggregate end-user traffic coming from the DSL connections and it was delivered to Edge IP routers. The fiber rings had Synchronous Digital Hierarchy (SDH) equipment as the standard transport technology for optical systems. There are still networks operating this way. However, current deployments are based on an IP/MPLS layer, either using dark fiber or optical switching network (WDM, OTN, etc.). This approach helps the operator to take advantage of the statistical multiplexing at the IP/MPLS layer, while the optical transmission improves the fiber utilization and provides a massive capacity.

The explosion of landline and mobile broadband has imposed an unprecedented traffic growth in telecommunication networks, with very high cumulative annual growth rates. Recently, Telefonica has seen how its number of customers has not increased at the same speed than the traffic demand. There are two main reasons: the economic crisis and the competition. Both factors have pushed Telefonica to do a huge investment in fiber infrastructure in order to speed up the deployment, while increasing the quality of the access portfolio. This factor has clearly impacted in the fiber access evolution in Spain (Fig. 1), where the million users goal was exceeded in 2014. This value was duplicated in 2015, moving from 1.590.990 users in 2014, to 3.161.302 in 2015. This huge increment of the access technology impacts on the capacity of the backbone networks, thus justifying the optimization of the IP/WDM technologies to maintain the network costs.

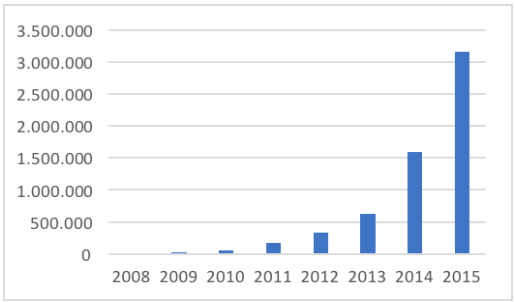



Fig. 1 FTTH users in Spain [1].

This paper is organized as follows: Section II introduces existing Open Source Frameworks, Section III explains the Netphony architecture and its components. Section IV presents some use cases validated with the platform. Section V demonstrates the architecture working with the Transport API as solution for SDN environments. Finally, Section VI concludes this paper.

## II. Existing Open Source frameworks

This section presents the most known Open Source frameworks, which can provide multi-layer control. There is a survey in [2] which provides a comprehensive and detailed analysis of the available SDN frameworks. There exists a large amount of SDN frameworks, which support different types of north- and south-bound APIs, different languages (C, Python, Java, etc.), support for consistency checks and fault tolerance, licensing scheme, and performances/scalability.

### *A. OpenDaylight*

OpenDaylight is an open source platform maintained by the Linux Foundation which provides network programmability to enable SDN and NFV for networks at any size and scale [3]. OpenDaylight enables SDN through a combination of components including a fully pluggable SDN controller, open interfaces, multi-protocol plug-ins and applications. The core part of the project, the modular, pluggable and flexible SDN controller is implemented in Java, while the northbound and southbound interfaces have clearly defined and documented APIs. The platform supports both OSGi framework and bidirectional REST for its northbound APIs: while OSGi is used for applications that will run in the same address space as the controller, REST APIs are used for applications that do not run in the same address space (or even necessarily on the same machine) as the SDN controller. Most important, OpenDaylight is widely supported by both the industry and the open source community; wide plethora of developers and users participates to the development and consolidation of this open source platform that it is licensed under the Eclipse Public Licence EPL-1.0.

### *B. ONOS*

Open Networking Operating System (ONOS) [4] is a large open-source effort to develop an SDN controller supported by a number of large companies. Like ODL, it is based on a Java-OSGI framework, with network-facing south-bound modules abstracting away unnecessary details and presenting a uniform view to higher layers. Its code is designed to be modular, configurable, protocol-agnostic, and expandable. This helps to support the configuration of the underlying networks using any protocol stack. ONOS is a multi-layer SDN control plane, manage both packet and optical layer networks, with multiple interfaces, equipment types, and protocols.

### *C. Open Source PCE*

The Open Source PCE is the first Open Source emulator of the PCE architecture [5]. Developed at Institut für Datentechnik und Kommunikationsnetze, TU Braunschweig, the PCE Emulator provides a framework for testing PCE capabilities in real network environments, fully extensible and designed to facilitate and enhance PCE research. The PCE Emulator is a free-to-use, licensed under the GNU GPL v3 license, and is publicly available for research and development use. The PCE emulator provides an extensible framework under which components of the PCE server and clients can be developed and extended in order to facilitate research and development activities in the PCE architecture. The Emulator has been developed in Java and currently consists of a complete PCE server implementation including protocol support according to RFC 4657, asynchronous network I/O, session management and support for extensible path computation and topology update mechanisms. The current framework provides support for basic path computation and session management, and efforts are on towards incorporating the full session management features along with standardized topology representation models inside the PCE.

The PCE Emulator has been developed in Java and is compatible with Java v1.6. The emulator was designed with extensibility in mind and as a result uses a modular architecture, which segregates the implementation of the major components of the PCE.

### *D. DRAGON*

Dynamic Resource Allocation in GMPLS Optical Networks (DRAGON) is a project funded by the National Science Foundation in the US, is focused on the provisioning of dynamic, deterministic end-to-end network transport for high-end applications on an inter-domain basis and across heterogeneous network topologies [6]. Although it does not consider the communication between the IP and carrier-grade management ecosystems, DRAGON supports UNI, so it can provision across network boundaries and differing network technologies with authentication, authorization, and accounting.

## III. Netphony Archictecture

The Netphony [8] is based on the IETF Application-Based Network Operations (ABNO) architecture [7]. The purpose of this architecture is to facilitate different implementations while offering interoperability between implementations of key components, and easy interaction with the applications and with the network devices. An implementation of this architecture may make several important decisions about the functional components. Multiple functional components may be grouped together into one software component. For example, an Active, Stateful PCE could be implemented as a single server combining the ABNO components of the PCE, the Traffic Engineering Database, the Label Switched Path Database, and the Provisioning Manager. These components could be distributed across separate processes. There could be multiple ABNO Controllers, each with capability to support different classes of application or application service.

The Netphony project is split in some libraries in other to facilitate the integration and extension with other frameworks. Following sections presents the modules and its functionality. The references to each repository can be found in the project's wiki [8].

### *A. Networking Protocol Library*

This library contains the java implementation of the main networking protocol stacks that enable the network control plane functions: PCEP protocol, RVP-TE protocol, OSPF-TE and BGP-LS. This protocol library can be integrated by any software requiring to use of these protocols.

### B. *Networking Emulator Library*

This repository contains a reference Java implementation of a Transport Network Emulator. It includes the emulator of a transport Node with GMPLS control capabilities. The Transport Node Emulator has a RSVP-TE process and an OSPF-TE daemon, as well as PCEP interface for remote instantiation of LSPs. The PCEP interface is stateful, so it reports its LSPs through that interface. The emulator is also distributed packaged in a virtual machine that can be found at [8].

### C. *Topology Module*

The next library has two main features, a Traffic Engineering Database (TEDB) and a BGP-LS Peer. The TEDB stores in memory a graph with nodes, links and their traffic engineering attributes. The netphony-topology library also includes a BGP-LS speaker (BGP-LS peer). The network-protocol library only included the protocol encoding, but not actually an executable. BGPPeer is the reference implementation of the BGP-LS speaker, acting as a BGP4 peer, it initiates BGP connections with its designated peers and, simultaneously, waits for incoming connections. The topology module can import the topology using other protocols like OSPF.

### D. *Path Computation Element (PCE)*

The PCE is defined in [9], and it is the unit that handles the path computation across the network graph. It can calculate traffic engineered end-to-end paths in order to optimize the optical spectrum consumption within the network. The PCE is capable of computing a TE LSP by operating on the TED regarding to the available bandwidth and network constraints. Coordination between multiple PCEs operating on different TEDs is also required for performing path computation in multi-domain (for example, inter-AS) or multi-layer networks.

This repository contains two implementations of a Java based Path Computation Element, a domain PCE and a Parent PCE. The repository also contains two Path Computation Clients, QuickClient, which by means of command line options can generate and receive PCEP messages.

### E. *Netphony ABNO*

This repository contains a reference implementation of the main components of the ABNO Architecture [7]. In particular, the main components are the ABNO controller and Provisioning Manager. The PCE is located the previous repositories. Following, the two modules are introduced.

The ABNO Controller is the main component of the architecture and is responsible of orchestrating, and invokes the necessary components in the right order. It listens for request from the NMS/OSS and selects the appropriate workflow to follow in order to satisfy each request.

The Provisioning Manager is the unit in charge of configuring the network elements so the LSP can be established. It can do so both by configuring the resources through the data plane or by triggering a set of actions to the control plane.

There are several protocols that allow the configuration of specific network resources such as Openflow, Netconf, CLI and PCEP.

### F. *tNetwork graphical library*

TID released a JavaScript visualization library to create and draw network graphs. It is SVG-based and HTML 5 compatible. This is a free library publicly available on GitHub, licensed under the GNU Affero license. It is available at the following location: https://github.com/telefonicaid/cne-tnetwork. This library is a complement of the Netphony framework and it allows to show the topological information. Fig. 2 shows an example of Telefonica of Spain reference network using the GMPLS control plane from Netphony.

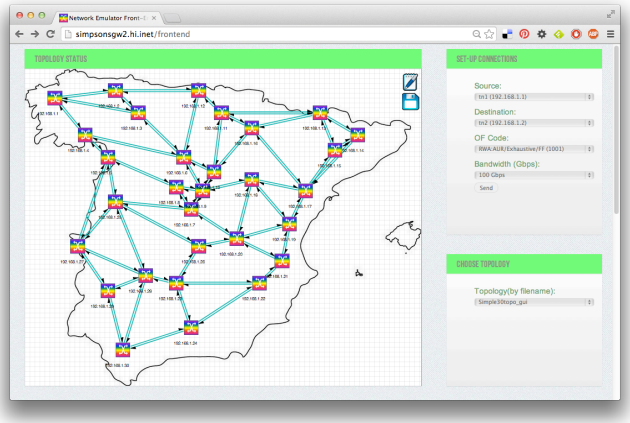



Fig. 2 tNetwork utilization to display the GMPLS control plane

## IV. END-TO-END ORCHESTRATION

Distributed computing capacity is becoming the common norm for efficient service provision by a telco operator, not only for external customers with offerings based in cloud (like storage, virtual machines, etc) but also for internal purposes (for instance, exploring the capabilities of virtualizing and distributing existing network functionalities on core platforms for fixed and mobile services). In such cases, to keep the latency to a minimum is crucial to guarantee a service experience as in a non-distributed case, and then optical transport technologies become a key part of the solution.

This distributed environment usually traverses more than a single network, and even within a network, more than one single domain. In order to compose and end-to-end service, orchestration capabilities are needed at least for configuring each edge device for every domain. When the services pass through multiple domains the number of manual interventions increases, and this gets even worst considering multiple technologies in the network (as is the case of distributed computing capabilities connected by means of optical equipment).

The solution addressed in the project to automate inter-site connectivity over different network segments (metro, core, data-center) is based on the combination of distribute control plane protocols and centralized SDN concepts.

Control plane protocols reduce the complexity of the building blocks in the ABNO, while centralized entities provide the orchestration and centralized view that are required in some steps of the operation. The centralized entities are clear thanks to the ABNO architecture. Regarding distributed technologies there are two candidates for different scenarios: Seamless MPLS and GMPLS.

- Seamless MPLS. Seamless MPLS is a multidomain solution, which can be used to create services in multiple ASes. In the case of provisioning a Seamless service, the ABNO architecture can be used to do just a request to the ABNO controller and the configuration can be done automatically without more human interaction.
- GMPLS. GMPLS is a set of protocols that enables the distributed computation and establishment of connections in the network. From a transport point of view, this set of protocols facilitates the procedures in the network in multidomain environments. In the case of optical networks, the ABNO architecture can be used to do a remote LSP instantiation as proposed in [6] to set-up an LSP in the network.

Although these protocols enables the end-to-end provisioning of services in certain technological islands. When there are multiple technologies in the network, the number of interventions can increase. Let us assume that some services are virtualized, so there is an OF domain for data center and pseudowire is terminated in a virtual AN, as depicted in Fig. 3. In this case, ABNO has to compute the E2E path and to carry out the configuration not only the PEs, but also Ethernet switches within the datacenter.

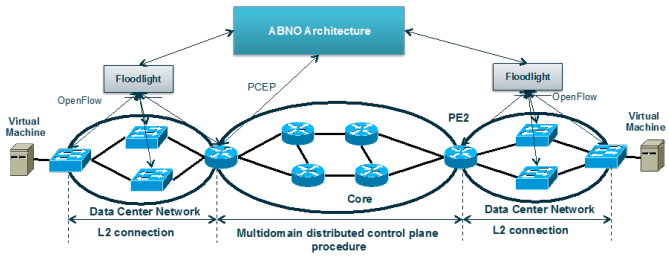



Fig. 3 End-to-end orchestration use case

## V. Workflow to Validate the Use Case

The following steps are required to provision bandwidth using the ABNO:

0. The initial trigger for this use case is a request from an end user via an NMS or an application.
1. The NMS sends the information to the ABNO controller with the two switches identifiers (Switch1 and Switch2) and the traffic parameters (e.g. bandwidth) required for the circuit.
2. When the ABNO controller receives the request, it identifies if this is a L2 or a L2 request from the parameters. In this case, the ABNO asks to the L2 PCE for a path with enough bandwidth in the L2 topology. If there is not a path in the L2 topology a NO PATH is sent to the ABNO. If there is a path move to step 10.
3. As there is no path or not enough bandwidth in the L2 topology, the ABNO controller tells VNTM to create a link between the two switches (Switch1 and Switch2).
4. The VNTM requests the L0 PCE a path between the two optical nodes connected to the two switches. If the L0 PCE is able to find it, it sends the path back to the VNTM.
5. VNTM has the interlayer information and the L0 path. This information is sent to the Provisioning Manager to configure the link.
6. The Provisioning Manager queries the Topology Module for the description of each node.
7. Depending on the technology and configuration mode, Provisioning Module selects a different protocol to complete the request. There are some options: PCEP, CLI to the router and UNI or even OF. For this demo, PCEP is used to do a remote LSP instantiation in the GMPLS nodes.
8. Once the path in the optical layer has been established, the Provisioning Manager notifies the VTNM. Similarly, VTNM notifies the ABNO controller.
9. ABNO controller asks the L2 PCE for a path with enough BW in the L2 topology. Now there is bandwidth to cope with the request, so the L2 PCE responds to ABNO with the path.
10. The ABNO controller requests the Provisioning Manager to configure the L2 service by establishing the path.
11. The Provisioning Manager queries the Topology Module for the description of each node.
12. Depending on the technology and configuration mode, Provisioning Module selects a different protocol to complete the request. To configure the IP layer there are some options: PCEP, CLI or NetConf. For this demo, a request is sent to a Floodlight controller, which configures the switches via OF.
13. Once the path has been established, Provisioning Manager notifies the VTNM.
14. Similarly, ABNO controller advertises the NMS.

Fig. 4 shows the steps in the modules of the ABNO architecture.

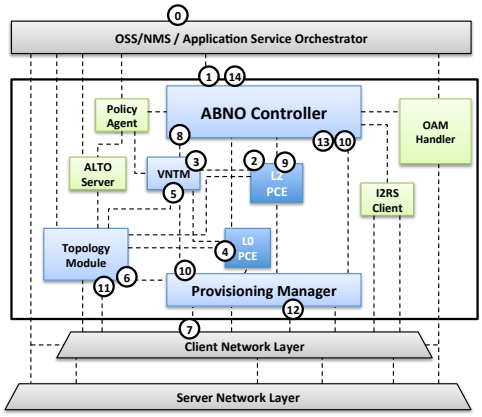



Fig. 4 Workflow for the E2E orchestration use case mapped in ABNO architecture

Fig. 5 explains this use case with a temporal workflow presenting the modules involved in each step and the actions done.

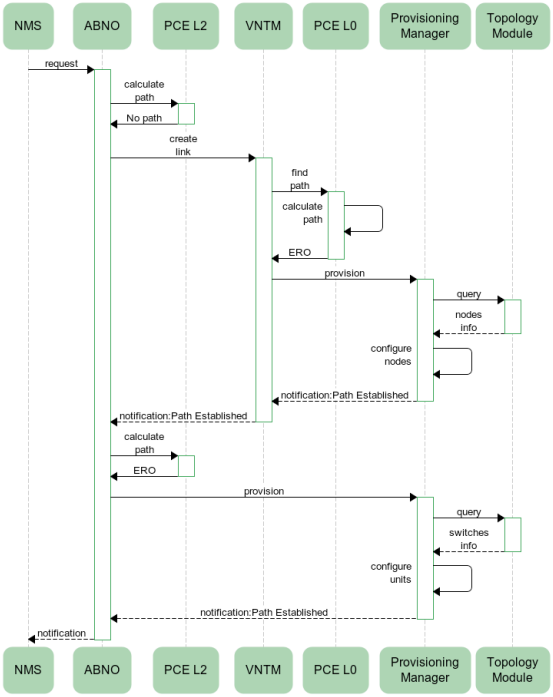



Fig. 5 Temporal workflow for the E2E orchestration use case

## VI. Experimental Validation

BGP-LS extends the BGP Update messages to advertise link-state topology thanks to new BGP Network Layer Reachability Information (NLRI). The Link State information is sent in two BGP attributes, the MP_REACH (defined in RFC 4670) and a LINK_STATE attribute (defined in the BGP-LS draft). To describe both the intra and inter domain links, in the MP_REACH attribute, we use a Link NLRI, which contains in the local node descriptors the address of the source, and in the remote descriptors, the address of the destination of the link. The Link Descriptors field has a TLV (Link Local/Remote Identifiers), which carries the prefix of the Unnumbered Interface. In case of the message informs about an intra-domain link, the standard traffic engineering information is included in the LINK_STATE attribute.

Fig. 6 shows a trace where there is a BGP speaker in 192.168.1.1 which is sending the BGP-LS to the IP 192.168.1.2. We captured the traffic of are two nodes (192.168.1.3 and 192.168.1.4) that are signalling OSPF information.

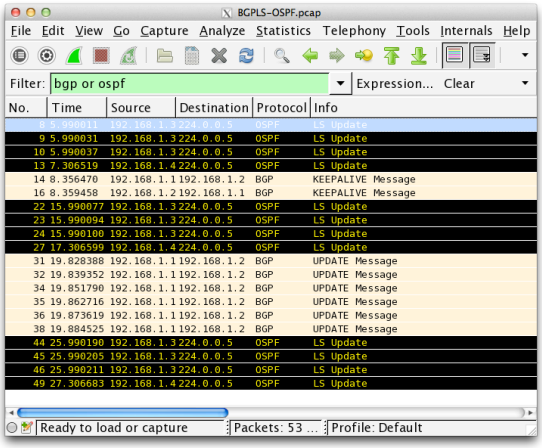



Fig. 6 BGP-LS message Exchange

Once the modules obtained the topology, the workflow can be executed. Fig. 7 shows the PCEP messages exchanged between the ABNO modules. Note that all the modules of the ABNO are running in the same physical host and the packets where captured in that host. This workflow is triggered based on the NMS request to the ABNO.

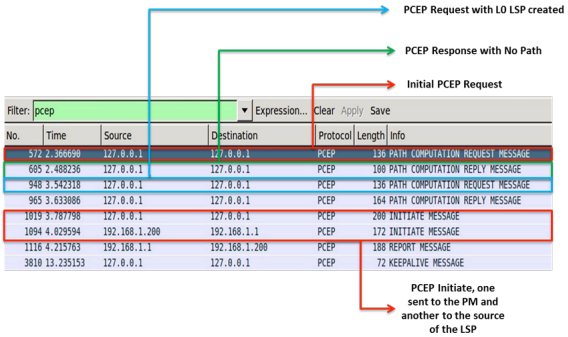
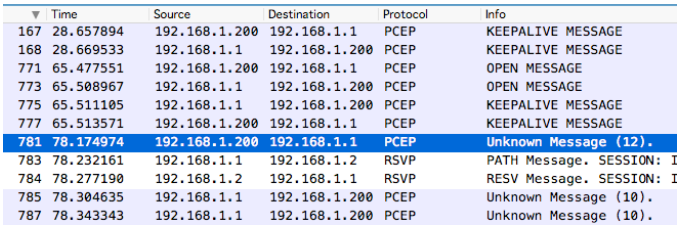



Fig. 7 PCEP messages exchanged between ABNO modules

When a PCInitiate message is received by the PM it asks the TM for information about each node. The TM gives for each IP in the ERO: the layer of the node, the configuration protocol (CLI, NetConf, OF or PCEP) and the specific parameters to configure the node. In the case of the optical layer it will reply with PCEP. This will make the PM to send a PCE message to the head end node, which will create a RSVP message with an ERO with the path. Fig. 8 shows the PCEP and RSVP messages that the first network element exchanges to setup a path.



| | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 167 | 28.657894 | 192.168.1.200 | 192.168.1.1 | PCEP | KEEPALIVE MESSAGE |
| 168 | 28.669533 | 192.168.1.1 | 192.168.1.200 | PCEP | KEEPALIVE MESSAGE |
| 771 | 65.477551 | 192.168.1.200 | 192.168.1.1 | PCEP | OPEN MESSAGE |
| 773 | 65.508967 | 192.168.1.1 | 192.168.1.200 | PCEP | OPEN MESSAGE |
| 775 | 65.511105 | 192.168.1.1 | 192.168.1.200 | PCEP | KEEPALIVE MESSAGE |
| 777 | 65.513571 | 192.168.1.200 | 192.168.1.1 | PCEP | KEEPALIVE MESSAGE |
| 781 | 78.174974 | 192.168.1.200 | 192.168.1.1 | PCEP | Unknown Message (12). |
| 783 | 78.232161 | 192.168.1.1 | 192.168.1.2 | RSVP | PATH Message. SESSION: I |
| 784 | 78.277190 | 192.168.1.2 | 192.168.1.1 | RSVP | RESV Message. SESSION: I |
| 785 | 78.304635 | 192.168.1.1 | 192.168.1.200 | PCEP | Unknown Message (10). |
| 787 | 78.343343 | 192.168.1.1 | 192.168.1.200 | PCEP | Unknown Message (10). |

Fig. 8 PCEP messages exchanged between ABNO modules

## VII. Conclusions

Network operators must deal with multi-layer architectures in their production networks. Current deployments are based on an IP/MPLS layer over an optical infrastructure (WDM, OTN, etc.). Both layers allow to take advantage of statistical multiplexing and achieve unprecedented capacity thanks to the underlying layer.

This multi-layer architecture requires a network programmability layer that enables the control and management of the IP and the optical resources. Our approach of multi-layer network programmability contains a hybrid approach to lever on the advantages of both paradigms. In terms on network protocols, there are to main trends binary versus REST-based. Our view is that binary protocols improve the performance at the low levels on the network, while REST-based APIs enables a faster development and network interoperability.

The Open Source Netphony suite is composed by a GMPLS control plane to emulate the network elements control, a Path Computation Element with active and stateful capabilities, a Topology Module capable of importing and exporting TE information in different protocols as well as an Application-based Network Operations (ABNO) controller. This framework enables multi-layer programmability for IP and optical networks.

## Acknowledgments

The work leading to the Netphony Orchestration suite comes from a set of EU Research Initiatives Telefonica Innovation program and PhD Thesis. This work was partially supported by ACINO European H2020 project, Grant Number 645127, http://www.acino.eu .

## References

[1] Comisión Nacional de los Mercados y la Competencia, “Annual Report 2016”, 14-10-2016, http://data.cnmc.es/

[2] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[3] OpenDayLight, http://www.opendaylight.org/

[4] ONOS, http://onosproject.org/

[5] Open Source PCE, https://www.tu-braunschweig.de/kns/projects/opensourcepce/

[6] DRAGON, http://cnl.gmu.edu/dragon/

[7] D. King and A. Farrel, RFC7491 “A PCE-Based Architecture for Application-Based Network Operations”.

[8] Netphony Suite https://github.com/telefonicaid/netphony-abno/wiki

[9] A. Farrel et al. “RFC4655 - A Path Computation Element (PCE)-Based Architecture”.

[10] 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.