Transport API: A Solution for SDN in Carriers Networks

Victor Lopez(1), Ricard Vilalta (2), Victor Uceda(1), Arturo Mayoral(2), Ramon Casellas (2), Ricardo Martínez(2), Raul Muñoz (2), Juan Pedro Fernandez Palacios (1)

(1) Telefónica I+D/GCTO (victor.lopezalvarez@telefonica.com), (2) CTTC (ricard.vilalta@cttc.es)

Abstract *The ONF Transport API is an interface to enable control of Transport networks, including services such as topology, or connectivity setup. We present the first demonstration of a connectivity service over a DWDM network using the ONF Transport API.*

Introduction

For several years, network operators have worked in get interoperability between multiple transport network scenarios, in order to encourage the competition among equipment manufacturers and to have more efficient network solutions. Nowadays, the increment of bandwidth requirements in the transport network, in addition to the actual environment where the end-user is willing to pay less for the service, force a decrease of revenues.

Software Define Networks (SDN) architecture allows the network operators to have multi-vendor interoperability. This architecture offers automated and simplified network service provisioning through different vendors, network segments (e.g., metro, core, data center) and technologies (e.g., IP/MPLS, optical, OpenFlow). The main idea of Software Define Network (SDN) is to separate the control and data plane allowing network programmability and this is especially appropriate in optical network, which signalling is done always via an out of band channel. In the previous model the controller was a NMS and there was no standard way to communicate the end devices with the NMS. The utilization of open and standard interfaces to enable interoperability is the first advantage of this SDN architecture.

Network operators usually acquire product from different vendors, because they use multiple technologies coexisting in their networks. Also multi-domains networks are required for the operators in order to cope with administrative and regional organizations. However, the current solutions in the market for SDN are limited, only based on single domain and mono vendor solutions. A single SDN controller presents scalability and reliability problems when we try to configure the whole network of an operator. These problems are compounded when considering an architecture that should deal with multiple South Bound Interfaces (SBI) like OpenFlow and GMPLS.

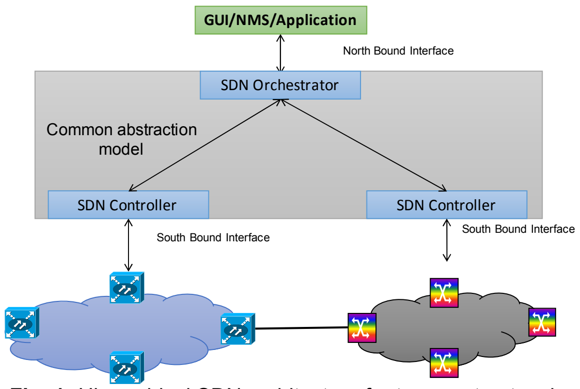


Fig. 1: Hierarchical SDN architecture for transport networks

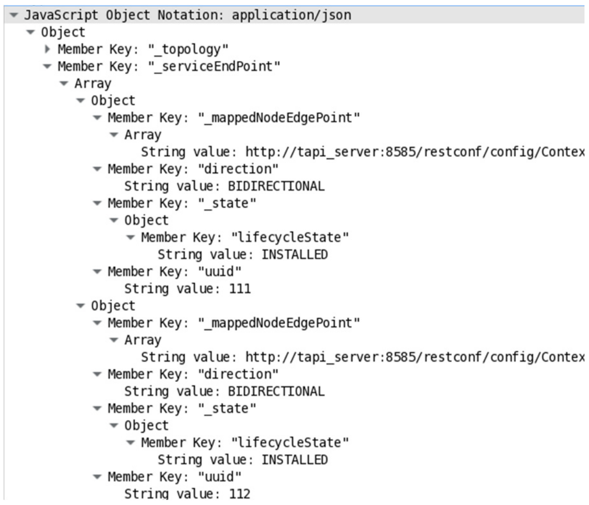
The Open Networking Foundation (ONF) has proposed a hierarchical SDN architecture that fits with the multi-vendor/multi-domain scenario. In this approach, multiple SDN controllers interact with an SDN orchestrator hierarchically placed on top of them, as shown in Fig.1.

In Transport Networks, where network-control function and behavior are well-understood and established, standardizing application programmer's interfaces (APIs) to the network control functions becomes important. ONF is reviewing the set of functional requirements and information model for the Transport API[2] (T-API), while implementing in SNOWMASS OpenSourceSDN.org project the data models for it (T-API YANG models)[3].

This paper extends the previous work in[1] analysing the Transport API capabilities and achieving the first experimental demonstration of a connectivity service over a DWDM network using the ONF Transport API over RESTconf protocol.

Transport API for Carrier SDN scenarios

The T-API abstracts the main services for an SDN controller: (1) Network Topology, (2) Connectivity Requests, (3) Path Computation, (4) Network Virtualization, and (5) Notification.



```
▾ JavaScript Object Notation: application/json
  ▾ Object
    ▸ Member Key: "_topology"
    ▾ Member Key: "_serviceEndPoint"
      ▾ Array
        ▾ Object
          ▾ Member Key: "_mappedNodeEdgePoint"
            ▾ Array
                String value: http://tapi_server:8585/restconf/config/Contex
          ▾ Member Key: "direction"
              String value: BIDIRECTIONAL
          ▾ Member Key: "_state"
            ▾ Object
              ▾ Member Key: "lifecycleState"
                  String value: INSTALLED
          ▾ Member Key: "uuid"
              String value: 111
        ▾ Object
          ▾ Member Key: "_mappedNodeEdgePoint"
            ▾ Array
                String value: http://tapi_server:8585/restconf/config/Contex
          ▾ Member Key: "direction"
              String value: BIDIRECTIONAL
          ▾ Member Key: "_state"
            ▾ Object
              ▾ Member Key: "lifecycleState"
                  String value: INSTALLED
          ▾ Member Key: "uuid"
              String value: 112
```

Fig. 2: T-API context JSON

Network Topology functionality requires, at a minimum, that the interface exports the context (Fig.2), which is the scope of control, interaction and naming that a particular T-API provider (SDN controller) or client application has with respect to the information exchanged over the interface. The context describes the Service End Points, which refer to the outward customer-facing aspects of the edge-port functions, and the Network Topology. However, network identifiers help to carry out path computation and to integrate the nodes for an end-to-end scenario. Further, the controllers can provide information about the links and nodes in the domain (complete or abstracted, depending on the configured shared context). It is clear that the more information is shared, the less abstracted the network appears. The Connectivity Service (Fig.2.d) represents an "intent-like" request for connectivity between two or more Service-End-Points. Instead, a connection is the provisioned potential for forwarding (circuit/packet) between two or more Node Edge Points of a Node. However, there are other Connectivity Constraints that can be requested, such as (a) excluding or including nodes/links for traffic engineering, (b) defining the protection level, (c) defining its bandwidth or (d) defining its disjointness from another connection.

The Path Computation function is a critical and fundamental feature because individual controllers in each domain are only able to share abstracted information that is local to their domain. An orchestrator with its global end-to-end view can optimize end-to-end connections that individual controllers cannot configure.

Network Virtualization services enables to expose a subset of the network resources to different tenants.

Finally, a Notification service allows a publish

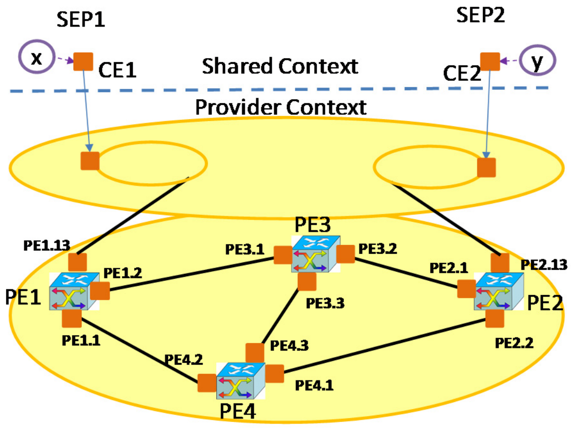


Fig. 3: Shared and Provider context views

and subscribe mechanism, which will allow asynchronous notifications using a protocol such as websockets.

In view of this, we can affirm that the T-API is in a good position to be the NBI of the SDN orchestrator or as the common abstraction model between the SDN orchestrator and the controllers. The importance of T-API relays in its simplicity and usability in order to extend its adoption. Other solutions based on YANG models are being proposed in the scope of IETF and OpenConfig.

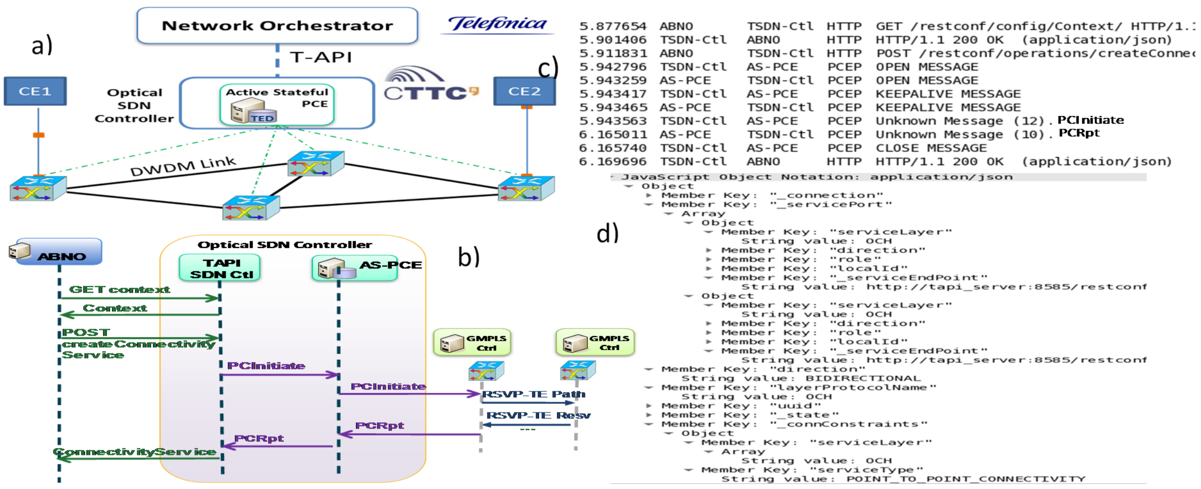
Connectivity service provisioning use case

Fig.3 shows both the shared view and the provider view of the proposed use case. In this scenario, there is no information about an abstract network topology shared between the client application and service provider. In this case, only the service end points are shared knowledge. The Provider view displays the different relationships between the service end points and customer edges and provider edges.

Only the Topology (to recover the context, which includes the service end points) and Connectivity service APIs are used by the client to manage connectivity services between service end points. No path constraints can be requested in the connectivity setup request and no path information can be returned for a connection.

When a connectivity service request T-API is received, a Transport SDN controller within the service provider will internally call its path computation to setup the connection within the service provider network.

In this use case the client is willing to dynamically create a OCh between two of its CE connected to the SP network through DWDM links. In order to support this use case, it is enough to pre-configure two service end points: X, and Y such that, to create the OCh between CE1 and CE2, a connectivity service needs to be requested between X and Y.



```
5.877654 ABNO      TSDN-Ctl HTTP GET /restconf/config/Context/ HTTP/1.
5.901406 TSDN-Ctl  ABNO     HTTP HTTP/1.1 200 OK  (application/json)
5.911831 ABNO      TSDN-Ctl HTTP POST /restconf/operations/createConne
5.942796 TSDN-Ctl  AS-PCE   PCEP OPEN MESSAGE
5.943259 AS-PCE    TSDN-Ctl PCEP OPEN MESSAGE
5.943417 TSDN-Ctl  AS-PCE   PCEP KEEPALIVE MESSAGE
5.943465 AS-PCE    TSDN-Ctl PCEP KEEPALIVE MESSAGE
5.943563 TSDN-Ctl  AS-PCE   PCEP Unknown Message (12). PCInitiate
6.165011 AS-PCE    TSDN-Ctl PCEP Unknown Message (10). PCRpt
6.165740 TSDN-Ctl  AS-PCE   PCEP CLOSE MESSAGE
6.169696 TSDN-Ctl  ABNO     HTTP HTTP/1.1 200 OK  (application/json)
```

```
JavaScript Object Notation: application/json
  Object
    Member Key: "_connection"
    Member Key: "_servicePort"
      Array
        Object
          Member Key: "serviceLayer"
            String value: OCH
          Member Key: "direction"
          Member Key: "role"
          Member Key: "localId"
          Member Key: "_serviceEndPoint"
            String value: http://tapi_server:8585/restconf
        Object
          Member Key: "serviceLayer"
            String value: OCH
          Member Key: "direction"
          Member Key: "role"
          Member Key: "localId"
          Member Key: "_serviceEndPoint"
            String value: http://tapi_server:8585/restconf
    Member Key: "direction"
      String value: BIDIRECTIONAL
    Member Key: "layerProtocolName"
      String value: OCH
    Member Key: "uuid"
    Member Key: "_state"
    Member Key: "_connConstraints"
      Object
        Member Key: "serviceLayer"
          Array
            String value: OCH
        Member Key: "serviceType"
          String value: POINT_TO_POINT_CONNECTIVITY
```

Fig. 4: a) Network Scenario, b) Message exchange workflow, c) Wireshark capture, d) Connectivity Service JSON

Experimental demonstration

Fig.4.a shows the experimental network scenario, including a Network Orchestrator, from Telefónica, based on the Open Source Netphony ABNO[5] and an optical SDN controller (provided by CTTC) which handles the DWDM network of the ADRENALINE Testbed.

A T-API SDN controller has been automatically generated using the same procedure as described in[4]. The T-API SDN controller translates the requested T-API connectivity service towards an AS-PCE.

The message exchange between the Client (Network Orchestrator) and the optical SDN controller is shown in Fig.4.b. First, we observe that the client requests the context to the SDN controller. Once the service end points have been considered, a connectivity service is requested through a RESTconf remote procedure call. The internal autogenerated T-API SDN controller translates this request into the necessary PCInitiate command sent to the AS-PCE, which triggers an LSP connection. Once the connection has been established, the AS-PCE answers with a PCRpt, which is translated into the acknowledgement of the creation of the connectivity service, which includes a connectivity service JSON (Fig.4.d).

Fig.4.c shows the wireshark captures of the message exchange between the client (ABNO), the T-API SDN Controller (TSDN-Ctl), and the AS-PCE. We can observe the different Uniform Resource Identifiers (URI), which are related with the RESTconf protocol using the defined TAPI data models (YANG).

Finally, Fig.4.d shows an excerpt of the connectivity service JSON, which includes the requested service ports, connection details, direction, and status, as well as the requested connectivity constraints.

Open issues within the T-API

Although the T-API is an important step forward towards a multi-vendor inter-operability solution for carrier networks, several issues remain open with the current release. The first issue is related to extendibility of the information model, towards an enhanced model for optical details, as well as other technologies as microwave[1]. The second issue is the alignment of the information model with IETF and other SDOs. Both issues are currently being handled.

Conclusions

This paper presents the Transport API as a solution for SDN Carrier networks as well as carries out the first experimental demonstration of a connectivity service over a DWDM network using the ONF Transport API data model and RESTconf protocol.

Acknowledgements

This research was partially funded by the European Commission H2020 Research and Innovation program, ACINO project, Grant Number 645127, www.acino.eu and EU FP7 project STRAUSS (FP7-ICT-2013-EU-Japan 608528) and Spanish MINECO project DESTELLO (TEC2015-69256-R). The authors would also like to thank the ONF Transport API design team.

References

[1] V. Lopez et al., “Towards a Transport SDN for Carriers Networks: An Evolutionary Perspective,” Proc. NOC, Lisbon (2016).

[2] K. Sethuraman (ed.), et al., “Functional Requirements for Transport API”, v.0.16, ONF.

[3] OpenSourceSDN.org Project SNOWMASS, https://github.com/OpenNetworkingFoundation/ONFOpenTransport/

[4] A. Mayoral, et al., First experimental demonstration of a distributed cloud and heterogeneous network orchestration with a common Transport API for E2E services with QoS, OFC 2016.

[5] Open Source Netphony ABNO, available at: https://github.com/telefonicaid/netphony-abno