Multi-domain Orchestration leveraging the Application-Layer Traffic Optimization Protocol

Danny Alex Lachos Perez[1], **Christian Esteve Rothenberg**[1]

[1] Universidade Estadual de Campinas (UNICAMP)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Information & Networking Technologies Research & Innovation Group (INTRIG)
Av Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Campinas, SP, Brasil

{dlachosp,chesteve}@dca.fee.unicamp.br

Abstract. *Evolving 5G network scenarios will have to deal with new multiple administrative domain (aka multi-domain) orchestration models. Based on a new broker-plane approach on top of per-domain management and orchestration functions, we present a Multi-domain orchestration prototype capable to coordinate the delivery of a multi-operator End-to-End Network Service (E2ENS) that combines per-domain paths and network functions for a given service request. Our prototype implementation retrieves local information about topology and resources from each Multi-domain Orchestrator (MdO) to offer after inter-domain added value services in the form of abstract Map Services. These map abstractions are easily consumable through developer-friendly REST APIs based on the Application-Layer Traffic Optimization (ALTO) standard protocol.*

1. Introduction

Future 5G network scenarios call for service orchestration model beyond the standard ETSI MANO. Complex multi-vendor, heterogeneous technology and resource environments consider broader collaboration mechanisms between different network operators. This deeper collaboration between different network operators is critical to enable new business model approaches, including federation models [5G-PPP 2013].

Presented in the IETF Application-Layer Traffic Optimization Working Group (ALTO WG), [Perez and Rothenberg 2018] proposes a federation networking paradigm where a broker-plane working on top of Multi-domain Orchestrators (MdOs) assists the coordinated creation of an End-to-End Network Service (E2ENS) spanning over multi-operator multi-domain networks. This proposed design resorts to the Application-Layer Traffic Optimization (ALTO) protocol [Alimi et al. 2014] to address the lack of abstractions to discover and adequately represent in confidentiality-preserving fashion the abstract network topology, resource availability (e.g., CPUs, Memory, and Storage) and capability (e.g., supported network functions) from different administrative domains.

Based on this brokerage level of orchestration, we implemented a Proof-of-Concept (PoC) framework, which point to the potential benefits of MdOs to be part of the federation and to be able to determine the underlying network graph and potential set of paths before bilateral negotiation between MdOs is started. In our prototype, each MdO involved in the federation advertises to the federation layer (acting as a broker) the intra-domain resource and topology information. From this local information, the broker

creates an aggregated inter-domain information exposed as a set of abstract and unified Map Services accessible to the MdOs through ALTO-based REST APIs.

The remainder of this paper is structured as follows. Section 2 provides an overview of the ALTO-based Broker-assisted MdO approach including the proposed architecture. A prototype implementation based on the aforementioned architecture is described in section 3. Section 4 validates the proof of concept with an initial functional analysis and discuss related work in Section 5. Finally, we conclude the paper in Section 6 and point to our future work.

2. ALTO-based Broker-assisted Multi-domain Orchestration

The approach towards our PoC use case implementation follows the design and development process of our ongoing work presented in IETF ALTO WG [Perez and Rothenberg 2018]. The primary design goal of this work is to discover resource and topology information from different administrative domains involved in a federation, while also safeguarding the privacy and autonomy of every domain.

The proposed brokered design is showed in Figure 1. In this reference architecture, the broker component is conceived to be working as coordinator of a set of MdOs. In turn, an MdO is assumed to manage a set of Domain Orchestrators (DOs) responsible for various resource domains (featuring physical and virtual, software and hardware components).

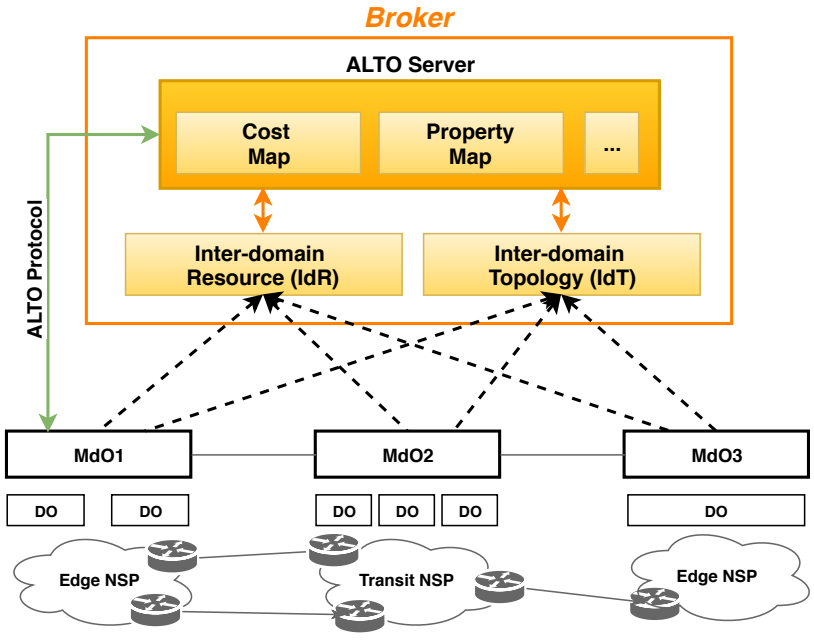

Figure 1. Broker-assisted Multi-operator Network Architecture

The main architectural components are described next:

2.1. Components

- **Inter-domain Resource (IdR):** It creates a hierarchical database that contains inter-domain resource information such as resource availability (i.e., CPU, mem-

ory, and storage), Virtual Network Functions (VNFs) and Physical Network Functions (PNFs) supported and Service Access Points (SAPs) to access those resources and VNFs/PNFs.
UNIFY [UNIFY D3.2a 2015], TOSCA [OASIS 2013], ETSI-NFV [ETSI 2014], among other data models can be used to create the interface between IdR and MdOs.

- **Inter-domain Topology (IdT):** A hierarchical TED (Traffic Engineering Database) that contains inter-domain network topology information including additional key parameters (e.g., throughput and latency of links). From this inter-domain TED information, can be created an aggregated domain-level topology map.
  The communication between IdT and MdOs components can be done using BGP-LS or REST interfaces.
- **ALTO Server:** The ALTO server component is the core of the broker layer. The information collected from the IdR and IdT modules are processed here to create and provide abstract maps with a simplified, yet enough information view about MdOs involved in the federation. This information includes domain-level topology, storage resources, computation resources, networking resources and PNF/VNF capabilities.
  As an ALTO client, each MdO sends ALTO service queries to the ALTO server. This server provides aggregated inter-domain information exposed as set ALTO base services defined in [Alimi et al. 2014], e.g., Network Map, Cost Map and ALTO extension services, e.g., Property Map [Roome et al. 2018], Multi-Cost Map [Randriamasy et al. 2017], Path Vector [Bernstein et al. 2018]. These ALTO extensions (specifically, the Property Map and Cost Map services) are introduced in the first version of the ALTO-based Multi-domain Orchestration IETF draft [Perez and Rothenberg 2018] with the goal to support the main functionalities in the proposed architecture.

3. Prototype Implementation

The strawman use case scenario refers to an E2ENS orchestration involving seven different administrative domains (3 Service Providers (SPs) and 4 Transit Providers (TPs)), as shown in Figure 2. In this section, we provide information about the implementation choices and prototype details such as the MdO components, the Neo4j[1] graph-based database used as the back-end for the ALTO information, and the OpenDaylight[2] (ODL) controller used as ALTO server.

3.1. MdO Components: 5GEx Project

The MdO functional components and interfaces follow the 5GEx project architectural proposal[3]. Each administrative domain has an MdO to manage resource and/or service orchestration at multi-operator level (via interface I2 APIs). Within the same administrative domain, each MdO uses emulated DOs (e.g., SDN, Mininet, Openstack, etc.) with emulated I3 interfaces, since no data-plane is present, i.e., DOs use static configuration files to load local information about topology (I3-RT) and resources (I3-RC).

[1] http://neo4j.com/
[2] https://www.opendaylight.org/
[3] http://www.5gex.eu/wp/?page id=510

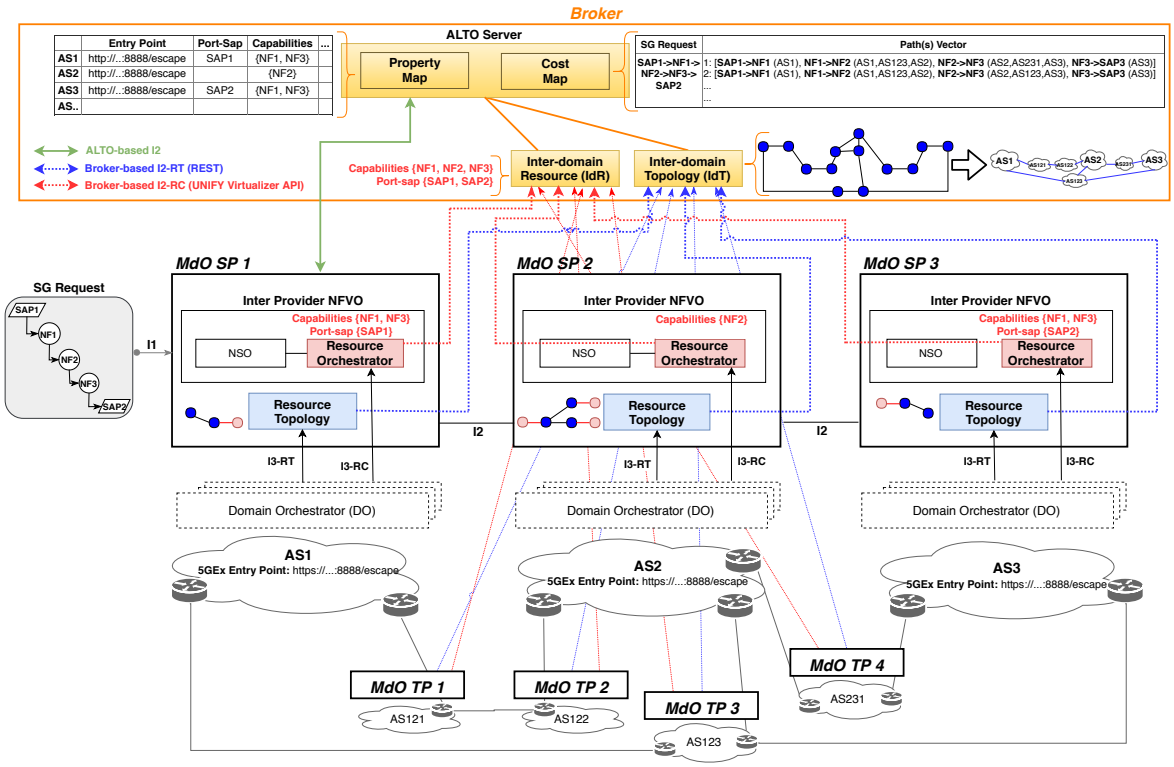

Figure 2. Multi-domain Orchestration Scenario

The different MdO components are based on existing open source software tools, for example, ESCAPE[4] and Netphony-topology[5] are used as Resource Orchestrator and Resource Topology, respectively. ESCAPE (Extensible Service ChAin Prototyping Environment) is a framework which supports the development of several parts of the service chaining architecture (e.g., VNF implementation, traffic steering, virtual network embedding, etc.) and it also includes a simple service layer interacting with clients. Netphony-topology is Java-based Traffic Engineering Database (TED) working as a BGP-LS Speaker that contains a Topology Module with a collection of TEDs and plugins to export and import the TEDs. Besides, MdOs expose I1 interfaces to the tenants who request services and/or slices which should follow a Network Function Forwarding Graph (NFFG) [UNIFY D3.2a 2015] format.

3.2. Broker Components

In case of the broker layer, the IdR and IdT components use the UNIFY Virtualizer API [UNIFY D3.2a 2015] (broker-based I2-RC API) and REST API (broker-based I2-RT API) respectively, to create the hierarchical databases. From the inter-domain information are created the two different ALTO Map Services: (i) *Property Map* and (ii) *Cost Map*.

- The **Property Map** includes property values grouped by Autonomous System (AS). Such values are SAPs, NFs, and the 5GEx Entry Point (e.g., the URL of the ESCAPE orchestrator).

[4]https://github.com/5GExchange/escape
[5]https://github.com/telefonicaid/netphony-topology

- The **Cost Map** defines a path vector as an array of ASes, representing the AS-level topological distance between entities (i.e., AS→AS, SAP→SAP, NF→NF or SAP↔NF). Moreover, as described in the Multi-Cost Map [Randriamasy et al. 2017], path vector constraints can be applied to restricts the response to costs that satisfy a list of simple predicates (e.g., $=$, $>$, $<$, $\geq$, $\leq$). Moreover, it is possible to use a special *"shortest"* predicate provide the shortest path between entities.

When an MdO receives a Service Graph (SG) request, it uses the ALTO server (through ALTO-based I2 APIs) to determine the underlying network graph and a potential set of paths before bilateral negotiation between MdOs is started.

3.3. Back-end/Front-end Servers

The resulting data for each broker component (IdR, IdT and ALTO server) is stored in Neo4j graph-based database (Back-end Server). We opt for this property graph[6] since it provides a natural modeling approach and it uses a key-value store abstraction for JSON object coding. Neo4j is an open-source non-relational graph database implemented in Java, and it supports true ACID transactions, high availability, and scales to billions of nodes and relationships [Neo4j 2015]. Moreover, its native traversal query language such as Cypher highly facilitates the development of applications.

The ALTO web server (Front-end Server) has been derived from the ALTO OpenDaylight (ODL) framework. The current release of ALTO[7] in ODL includes, among other modules, ALTO Northbound providing basic ALTO services as RESTful web services (Northbound APIs) for ALTO client/server communications. ALTO Northbound APIs generate ALTO services from data stored in the MD-SAL data store (an ODL core component). For our implementation, it was necessary to modify the Northbound APIs to generate ALTO services from the data stored in the Neo4j back-end and converts it into the ALTO format specification.

4. Experimental Evaluation

The initial evaluation for our testing environment (see Fig. 2) includes the function behavior. This means that we evaluate whether our ALTO server delivers ALTO services in compliance with ALTO base services defined in RFC7285 [Alimi et al. 2014], e.g., Network Map, Cost Map and ALTO extension services, e.g., Property Map [DRAFT-PM], Multi-Cost Map [RFC8189], Path Vector [DRAFT-PV]. For that purpose, we used a REST client tool[8] to retrieve ALTO information in JSON format, communicating with the ALTO server via HTTP request.

Examples of the Filtered Property Map and Filtered Cost Map queries and the corresponding responses are featured below.

4.1. Filtered Property Map Service

In this example, the ALTO client wants to retrieve the Property Map for PID entities with the *"unifyslor"* (or MdO entry-point), *"cpu"*, *"mem"*, *"storage"*, *"port"* and *"nf"*

[6] A graph where (i) vertices and edges can have any number of key/value properties, (ii) there can be many types of relationships between vertices and (iii) edges have a directionality.

[7] https://wiki.opendaylight.org/view/ALTO:Main

[8] https://www.getpostman.com/

properties. The PIDs entities are MdO SP1 (0.0.0.1), MdO TP3 (0.0.0.123), MdO SP2 (0.0.0.2) and MdO SP3 (0.0.0.3).

- HTTP Request

```
POST /controller/nb/v2/alto/filtered/propertymap/my-default-property-map
Host: 172.28.0.10:8181
Accept: application/alto-propertymapfilter+json,application/alto-error+json

{
    "pids": [ "0.0.0.1", "0.0.0.123","0.0.0.2", "0.0.0.3" ]
}
```

- HTTP Response

```
{
    "meta": {
        "vtag": {
            "resource-id": "my-default-property-map",
            "tag": "4VSt4OFTRMBdc5gHIuLGhKUBL4xMXsP8"
        }
    },
    "property-map": {
        "0.0.0.1": {
            "unifyslor": "https://172.25.0.10:8888/escape",
            "cpu": "50.0",
            "mem": "60.0",
            "storage": "70.0",
            "port": [ "SAP1", "SAP1211", "SAP1231", "SAP2" ],
            "nf": [ "COMPRESSOR", "DECOMPRESSOR" ]
        },
        "0.0.0.123": {
            "unifyslor": "https://172.52.0.10:8888/escape",
            "cpu": "0.0",
            "mem": "0.0",
            "storage": "0.0",
            "port": [ "SAP1231", "SAP1232", "SAP1233" ],
            "nf": []
        },
        "0.0.0.2": {
            "unifyslor": "https://172.26.0.10:8888/escape",
            "cpu": "10.0",
            "mem": "20.0",
            "storage": "30.0",
            "port": [ "SAP1221", "SAP1232", "SAP2311", "port-SAP4" ],
            "nf": [ "FORWARDER" ]
        },
        "0.0.0.3": {
            "unifyslor": "https://172.27.0.10:8888/escape",
            "cpu": "80.0",
            "mem": "90.0",
            "storage": "100.0",
            "port": [ "SAP1233", "SAP2312", "SAP3" ],
            "nf": [ "COMPRESSOR", "DECOMPRESSOR" ]
        }
    }
}
```

Appendix A gives another example of a fully Property Map query and the corresponding response.

4.2. Filtered Cost Map Service

The following example uses the Filtered Cost Map service to request the path vector for a given E2E requirement. The SG request information is composed of three NFs: (NF1)

"COMPRESSOR", (NF2) *"FORWARDER"*, (NF3) *"DECOMPRESSOR"* and, two SAPs (SAP1 and SAP3). Links connecting the NFs and SAPs (*"sg_links"* tag) are also included, followed by an E2E requirement (*"reqs"* tag) with information about the order in which NFs are traversed from SAP1 to SAP3.

Note that the request includes a constraint (*"constraints" : ["= 9"]*) in order to return just AS-level paths for which the number of AS hops in the E2E requirement is equal to 9.

- HTTP Request

```
POST /controller/nb/v2/alto/costmap/pv
Host: 172.28.0.10:8181
Accept: multipart/related, application/alto-costmap+json,
application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
    "cost-type" :{
            "cost-mode": "array",
            "cost-metric": "ane-path"
    },
    "constraints" : [ "= 9"],
    "sg" :{
          "nfs": [ "COMPRESSOR", "FORWARDER", "DECOMPRESSOR"],
          "saps": [ "SAP1", "SAP3" ],
          "sg_links": [ {
                  "id": 1,
                  "src_node": "SAP1",
                  "dst_node": "COMPRESSOR"
                },{
                  "id": 2,
                  "src_node": "COMPRESSOR",
                  "dst_node": "FORWARDER"
                },{
                  "id": 3,
                  "src_node": "FORWARDER",
                  "dst_node": "DECOMPRESSOR",
                },{
                  "id": 4,
                  "src_node": "DECOMPRESSOR",
                  "dst_node": "SAP3"
                }],
          "reqs": [ {
                "src_node": "SAP1",
                "dst_node": "SAP3",
                "sg_path": [ 1, 2, 3, 4 ]
            }
          ]
    }
}
```

- HTTP Response

  For each SG link in the E2E requirement (SAP1->COMPRESOR, COMPRESOR->FORWARDER, FORWARDER->DECOMPRESOR, DECOMPRESOR->SAP3), the ALTO server returns sub-arrays indicating potential candidate paths calculated as the AS-level topological distance corresponding to the amount of traversing domains. This AS-level distance is limited to 9 hops as defined by the HTTP request of the above example.

```
{
    "meta": {
        "vtag": {
            "resource-id": "my-default-property-map",
            "tag": "4VSt4OFTRMBdc5gHIuLGhKUBL4xMXsP8"
        }
    },
    "cost-map": {
        "SAP1": {
            "SAP3": {
                "SAP1": {
                    "COMPRESSOR": [
                        [ "0.0.0.1" ]
                    ]
                },
                "COMPRESSOR": {
                    "FORWARDER": [
                        [ "0.0.0.1","0.0.0.121","0.0.0.122","0.0.0.2" ]
                      ]
                },
                "FORWARDER": {
                    "DECOMPRESSOR": [
                        [ "0.0.0.2","0.0.0.123","0.0.0.3"],
                        [ "0.0.0.2","0.0.0.231","0.0.0.3"]
                    ]
                },
                "DECOMPRESSOR": {
                    "SAP3": [
                        [ "0.0.0.3" ]
                    ]
                }
            }
        }
    }
}
```


A second example of the Filtered Cost Map service requesting the AS-level topological distance with constraints [*"shortest"*] is given in Appendix B.

Based on the initial experimental results, the prototype under evaluation presents a set of benefits leveraging the proposed brokering layer: (i) avoid the distribution of topology and resource information in a peer-to-peer fashion (the more MdO-to-MdO interconnections, the larger the *"costs"* of distribution); (ii) allow domains without physical infrastructure (e.g., without BGP or BGP-LS instances) to advertise and learn (avoiding the deployment and configuration of per domain BGP peering points); and (iii) use the joint inter-domain topology information to pre-select a (sorted) list of candidate domains (reducing the number of redundant links along the path(s) in E2ENS requests).

5. Related Work

Current proposals for management and orchestration are intrinsically conceived for single administrative domain scenarios. The standard service orchestration model described in ETSI NFV MANO framework [ETSI 2014], for example, describes the use of orchestrator(s) working within a single administrative domain. The analysis of network service/resource orchestration across multiple administrative domains has begun to be addressed by ETSI in [ETSI 2018].

Existing open source projects sprint the multi-provider multi-domain orchestration challenges under different approaches. [Bernardos et al. 2015] aims to integrate mul-

tiple administrations and technologies through the collaboration between operators in the context of emerging 5G networking. Other studies, such as [Demchenko et al. 2013], addresses problems with multi-domain (Multi-operator/multi-technology) heterogeneous cloud-based applications integration, however, it is not focused on the provisioning of NFV-based cross-domain network services.

Closest to our efforts, [VITAL 2015][T-NOVA 2014] follow a centralized approach where each domain advertises its capabilities to a federation layer which will act as a broker. In order to avoid one network operator per country or regions, [Banchs et al. 2015] proposes the use of management and control into a single virtual domain. Also, the 5G-Transformer project [5G-Transformer 2017] is defining flexible slicing and federation of transport networking and computing resources across multiple domains. All such proposals introduce new business model approach, including a federation model among administrative domains where each network operator involved in the community advertises its abstracted capabilities to a broker (i.e., 3rd party).

6. Conclusion and Future Work

Evolving networking scenarios (e.g., 5G) require the provision of value-added services in multi-domain (multi-operator/multi-technology) environments. In this work, we designed and implemented a use case prototype inspired in an ALTO-based Broker-assisted approach through which single domains can describe their resource and network capabilities in an interoperable manner.

Presented initial experiments bring essential guidelines towards potential benefits to the challenges of multi-domain orchestration (e.g., lack of abstractions, scalability, and flexibility) by leveraging the map services and generality of the ALTO protocol. Finally, taking into account latest efforts made by IETF/IRTF, future activities include pursuing contributions to standardization along (i) the way MdOs describe their resource/network capabilities; and (ii) extensions and/or new services to the base ALTO protocol as necessary.

7. Acknowledgements

This work is supported by the Innovation Center of Ericsson S.A., Brazil (grant agreement UNI.62).

References

5G-PPP (2013). Advanced 5G Network Infrastructure for the Future Internet. https://5g-ppp.eu/wp-content/uploads/2014/02/Advanced-5G-Network-Infrastructure-PPP-in-H2020_Final_November-2013.pdf.

5G-Transformer (2017). 5G-Transformer – 5G Mobile Transport Platform for Vertical. http://5g-transformer.eu/.

Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and Woundy, R. (2014). Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285.

Banchs, A., Breitbach, M., Costa, X., Doetsch, U., Redana, S., Sartori, C., and Schotten, H. (2015). A novel radio multiservice adaptive network architecture for 5g networks. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, pages 1–5. IEEE.

Bernardos, C. J., Dugeon, O., Galis, A., Morris, D., Simon, C., and Szabó, R. (2015). 5g exchange (5gex)-multi-domain orchestration for software defined infrastructures. *focus*, 4(5):2.

Bernstein, G., Chen, S., Gao, K., Lee, Y., Roome, W., Scharf, M., Yang, Y., and Zhang, J. (2018). Alto extension: Path vector cost type. Internet-Draft draft-ietf-alto-path-vector-03, IETF Secretariat.

Demchenko, Y., Makkes, M. X., Strijkers, R., Ngo, C., and Laat, C. d. (2013). Intercloud architecture framework for heterogeneous multi-provider cloud based infrastructure services provisioning. *International Journal of Next-Generation Computing*, 4(2).

ETSI (2014). Network Functions Virtualisation (NFV) Management and Orchestration V1.1.1. http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.

ETSI (2018). Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains V3.1.1. http://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/028/03.01.01_60/gr_NFV-IFA028v030101p.pdf.

Neo4j (2015). The Neo4j Manual v2.3.0-M03. http://neo4j.com/docs/milestone/.

OASIS (2013). TOSCA specification. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf.

Perez, D. and Rothenberg, C. (2018). Alto-based broker-assisted multi-domain orchestration. Working Draft.

Randriamasy, S., Roome, W., and Schwan, N. (2017). Multi-cost application-layer traffic optimization (alto). RFC 8189, RFC Editor.

Roome, W., Chen, S., Randriamasy, S., Yang, Y., and Zhang, J. (2018). Unified properties for the alto protocol. Working Draft.

T-NOVA (2014). T-NOVA Project, Network Functions as a Service over Virtualised Infrastructures. http://www.t-nova.eu/.

UNIFY D3.2a (2015). NFFG Specification. http://www.fp7-unify.eu/files/fp7-unify-eu-docs/Results/Deliverables/UNIGY_D3.2a_NFFG%20Specification.pdf.

VITAL (2015). VITAL – VIrtualized hybrid satellite-TerrestriAl systems for resilient and fLexible future networks. http://www.ict-vital.eu/.

Appendix A Property Map Service

This HTTP request example corresponds to the full (unfiltered) Property Map. The ALTO server defines a GET-mode resource which returns the entire Property Map for all PID entities with the *"unifyslor"*, *"cpu"*, *"mem"*, *"storage"*, *"port"* and *"nf"* properties.

- HTTP Request

```
GET /controller/nb/v2/alto/propertymap/my-default-property-map HTTP/1.1
Host: 172.28.0.10:8181
Accept: application/alto-propmap+json,application/alto-error+json
```

- HTTP Response


```
{
    "meta": {
        "vtag": {
            "resource-id": "my-default-property-map",
            "tag": "4VSt4OFTRMBdc5gHIuLGhKUBL4xMXsP8"
        }
    },
    "property-map": {
        "0.0.0.1": {
            "unifyslor": "https://172.25.0.10:8888/escape",
            "cpu": "50.0", "mem": "60.0", "storage": "70.0",
            "port": [ "SAP1", "SAP1211", "SAP1231", "SAP2" ],
            "nf": [ "COMPRESSOR", "DECOMPRESSOR" ]
        },
        "0.0.0.121": {
            "unifyslor": "https://172.50.0.10:8888/escape",
            "cpu": "0.0", "mem": "0.0", "storage": "0.0",
            "port": [ "SAP1211", "SAP1212" ],
            "nf": []
        },
        "0.0.0.122": {
            "unifyslor": "https://172.51.0.10:8888/escape",
            "cpu": "0.0", "mem": "0.0", "storage": "0.0",
            "port": [ "SAP1212", "SAP1221" ],
            "nf": []
        },
        "0.0.0.123": {
            "unifyslor": "https://172.52.0.10:8888/escape",
            "cpu": "0.0", "mem": "0.0", "storage": "0.0",
            "port": [ "SAP1231", "SAP1232", "SAP1233" ],
            "nf": []
        },
        "0.0.0.2": {
            "unifyslor": "https://172.26.0.10:8888/escape",
            "cpu": "10.0", "mem": "20.0", "storage": "30.0",
            "port": [ "SAP1221", "SAP1232", "SAP2311", "port-SAP4" ],
            "nf": [ "FORWARDER" ]
        },
        "0.0.0.231": {
            "unifyslor": "https://172.53.0.10:8888/escape",
            "cpu": "0.0", "mem": "0.0", "storage": "0.0",
            "port": [ "SAP2311", "SAP2312" ],
            "nf": []
        },
        "0.0.0.3": {
            "unifyslor": "https://172.27.0.10:8888/escape",
            "cpu": "80.0", "mem": "90.0", "storage": "100.0",
            "port": [ "SAP1233", "SAP2312", "SAP3" ],
            "nf": [ "COMPRESSOR", "DECOMPRESSOR" ]
        }
    }
}
```


Appendix B Filtered Cost Map Service

In this Filtered Cost Map service, the ALTO server returns connectivity information for an SG request provided by the HTTP request example. This request includes a constraint predicate (*"constraints" : ["shortest"]*) so that, the ALTO server returns the shortest AS-level topological distance which meets the E2ENS requirement.

- HTTP Request

```
POST /controller/nb/v2/alto/costmap/pv
Host: 172.28.0.10:8181
Accept: multipart/related, application/alto-costmap+json,
application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json
{
    "cost-type" :{
            "cost-mode": "array", "cost-metric": "ane-path"
    },
    "constraints" : [ "shortest"],
    "sg" :{
          "nfs": [ "COMPRESSOR", "FORWARDER", "DECOMPRESSOR"],
          "saps": [ "SAP1", "SAP3" ],
          "sg_links": [ {"id": 1,
                  "src_node": "SAP1",
                  "dst_node": "COMPRESSOR"
                },{"id": 2,
                  "src_node": "COMPRESSOR",
                  "dst_node": "FORWARDER"
                },{"id": 3,
                  "src_node": "FORWARDER",
                  "dst_node": "DECOMPRESSOR",
                },{"id": 4,
                  "src_node": "DECOMPRESSOR",
                  "dst_node": "SAP3"
                }],
          "reqs": [ {
                "src_node": "SAP1",
                "dst_node": "SAP3",
                "sg_path": [ 1, 2, 3, 4 ]
            }
          ]
    }
}
```

- HTTP Response

```
{
    "meta": {
        "vtag": {
            "resource-id": "my-default-property-map",
            "tag": "4VSt4OFTRMBdc5gHIuLGhKUBL4xMXsP8"}
    },
    "cost-map": {
        "SAP1": {
            "SAP3": {
                "SAP1": {"COMPRESSOR": [
                        [ "0.0.0.1" ]]
                },
                "COMPRESSOR": {"FORWARDER": [
                        [ "0.0.0.1","0.0.0.123","0.0.0.2" ]]
                },
                "FORWARDER": {"DECOMPRESSOR": [
                        [ "0.0.0.2","0.0.0.123","0.0.0.3" ]]
                },
                "DECOMPRESSOR": {
                    "SAP3": [
                        [ "0.0.0.3" ]]
                }
            }
        }
    }
}
```