IETF 101 - ALTO WG

ALTO-based Broker-assisted Multi-domain Orchestration - 00

Danny Alex Lachos Perez
Christian Esteve Rothenberg
(University of Campinas, Brazil)

@ http://intrig.dca.fee.unicamp.br

Agenda

1. Multi-domain Orchestration
2. Broker-assisted Multi-domain Orchestration Approach
3. Required ALTO Extensions
    a. Property Map
    b. Filtered Cost Map

Multi-domain Orchestration

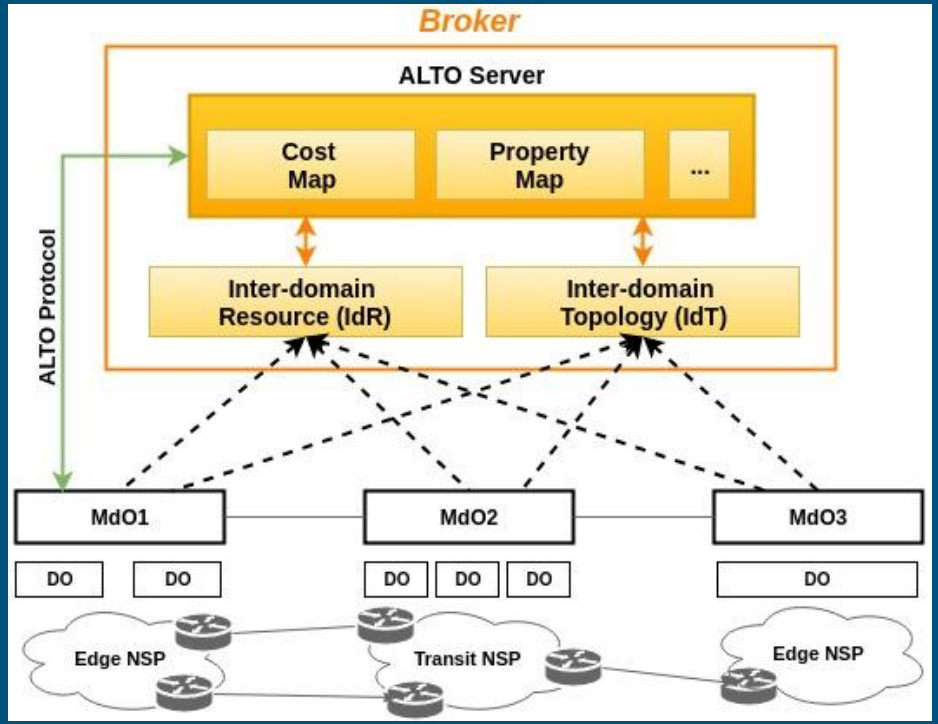
- 5G network scenarios call for multi-domain orchestration models.
- Multi-provider orchestration operations will require the information exchange across Multi-domain Orchestrators (MdOs).
- Information to be exchanged:
  - Abstract network topology
  - Resource availability (e.g., CPUs, Memory, and Storage)
  - IT Capabilities (e.g., supported network functions)
  - Orchestrator entry points
- Challenges:
  - Lack of abstractions
  - Discovery of candidate autonomous systems
  - Scalability, Flexibility, Complexity

Our Proposed Approach

- Proposal:
  - A federation networking paradigm where a broker-plane works on top of the management and orchestration plane.
- Main Goal:
  - Discover resource and topology information from different administrative domains involved in the federation.
- ALTO-based:
  - The ALTO services (with the proposed protocol extensions) offer abstract maps with a simplified view, yet enough information about MdOs involved in the federation.

Architecture

- Inter-domain Resource (IdR)
  - Resource availability
  - VNFs/PNFs
  - SAPs
- Inter-domain Topology (IdT)
  - Hierarchical TED
- ALTO Server
  - Property Map
  - Cost Map



Property Map Extensions

- The ALTO server MUST return multiple values for each property in the Property Map.
  - MdOs exchange a list NFs and SAPs which are supported by them. So in this scenario, an array of values can provide sufficient information that is not possible with single string values.

- Specifications (based on 4.6 of [DRAFT-PM]):
  - The specification for the "Media Types", "HTTP Method", "Accept Input Parameters", "Capabilities" and "Uses" remain unchanged.
  - **"Response" Specification:** For each property name defined in the resource's "capabilities" list, the corresponding property value MUST be encoded as JSONArray instead of JSONString.

Example: Property Map Service

- The ALTO client wants to retrieve the entire Property Map for PID entities with the "entry-point", "cpu", "mem", "storage", "port" and "nf" properties.

```
GET /propmap/full/inet-ucmspn HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
      "pid:AS1": {
          "entry-point": [ "http://172.25.0.10:8888/escape" ],
          "cpu": [ "50.0" ],
          "mem": [ "60.0" ],
          "storage": [ "70.0" ],
          "port": [ "SAP1" ],
          "nf": [ "NF1", "NF3" ]
      },
      "pid:AS2": {
          "entry-point": [ "http://172.26.0.10:8888/escape" ],
          "cpu": [ "10.0" ],
          "mem": [ "20.0" ],
          "storage": [ "30.0" ],
          "nf": [ "NF2" ]
      },
      "pid:AS3": {
          "entry-point": [ "http://172.27.0.10:8888/escape" ],
          "cpu": [ "80.0" ],
          "mem": [ "90.0" ],
          "storage": [ "100.0" ],
          "port": [ "SAP2" ],
          "nf": [ "NF1", "NF3" ]
      }
  }
}
```

Filtered Cost Map Extension (1/2)

- The ALTO server MUST provide connectivity information for every SG link in the SG path for an E2E requirement.
  - This information is the AS-level topological distance in the form of path vector, and it includes all possible ways for each (source node, destination node) pair in the SG link.

- Specifications (based on Section 6.1 of [DRAFT-PV]):
  - The specifications for the "Media Types", "HTTP method", "Capabilities" and "Uses" are unchanged.
  - **"Accept Input Parameters" Specification:** If “sg” is present, the ALTO Server MUST allow the request input to include an SG with a formatted body as an NFFG object.

```
object {
  [NFFG sg;]
} ReqFilteredCostMap;

object {
  JSONString nfs<1..*>;
  JSONString saps<1..*>;
  NextHops sg_links<1..*>;
  REQs reqs<1..*>;
} NFFG;

object {
  JSONNumber id;
  JSONString src-node;
  JSONString dst-node;
} NextHops;

object {
  JSONString id;
  JSONString src-node;
  JSONString dst-node;
  JSONNumber sg-path<1..*>;
} REQs;
```

Filtered Cost Map Extension (2/2)

- Specifications (based on Section 6.1 of [DRAFT-PV]):
  - **"Response" Specification:** If the ALTO client includes the path vector cost mode in the "cost- type" (or "multi-cost-types") field of the input parameter, the response for each SG link in each E2E requirement MUST be encoded as a JSONArray of JSONArrays of JSONStrings.
    - Moreover, as defined in Section 6.3.6 of [DRAFT-PV], If an ALTO client sends a request of the media type "application/alto-costmapfilter+json" and accepts "multipart/related", the ALTO server MUST provide path vector information along with the associated Property Map information, in the same body of the response.

Example: Filtered Cost Map (1/2)

- The ALTO client requests the path vector for a given E2E requirement:
  - SAP1->NF1->NF2->NF3->SAP2
- SG Request:
  - Three NFs (NF1, NF2, and NF3) .
  - Two SAPs (SAP1 and SAP2).
  - Four Links connecting the NFs and SAPs ("sg_links" tag).
  - An E2E requirement ("reqs" tag) with information about the order in which NFs are traversed from SAP1 to SAP2.
- Note:
  - The request accepts "multipart/related" media type. This means the ALTO server will include associated property information in the same response.

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
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"
  },
  "sg": {
    "nfs": [ "NF1", "NF2", "NF3" ],
    "saps": [ "SAP1", "SAP2" ],
    "sg_links":[
      {
        "id": 1,
        "src-node": "SAP1",
        "dst-node": "NF1",

      },
      {
        "id": 2,
        "src-node": "NF1",
        "dst-node": "NF2",
      },
      {
        "id": 3,
        "src-node": "NF2",
        "dst-node": "NF3",
      },
      {
        "id": 4,
        "src-node": "NF3",
        "dst-node": "SAP2",
      }
    ],
    "reqs": [
      {
        "id": 1,
        "src-node": "SAP1",
        "dst-node": "SAP2",
        "sg-path": [ 1, 2, 3, 4 ]
      }
    ]
  }
}
```

Example: Filtered Cost Map (2/2)

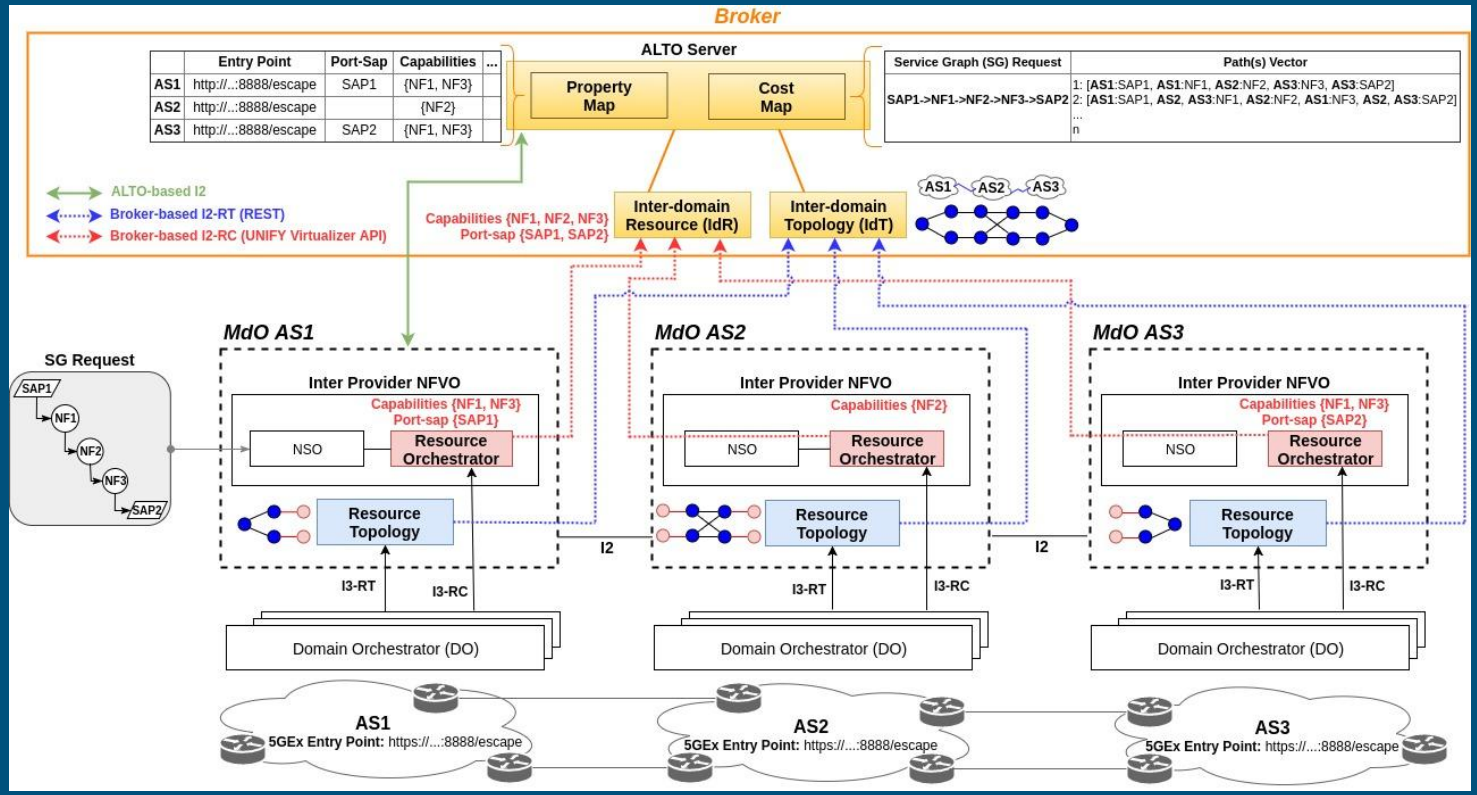
- The ALTO server returns connectivity information for the E2E requirement.
- The response includes Property Map information for each element in the path vector.
  - In this case, it is retrieved a Property Map with the "entry-point" property, i.e., the URL of the MdO entry point for the corresponding network.

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example

--example
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "cost-type": {
       "cost-mode": "array",
       "cost-metric": "ane-path"
     },
  },

  "cost-map": {
    "SAP1": {
      "SAP2": {
          "SAP1": {
              "NF1": [
                [ "AS1" ], [ "AS1", "AS2", "AS3" ]
              ]
          },
          "NF1": {
              "NF2": [
                [ "AS1", "AS2" ], [ "AS3", "AS2" ]
              ]
          },
          "NF2": {
              "NF3": [
                [ "AS2", "AS1" ], [ "AS2", "AS3" ]
              ]
          },
          "NF3": {
              "SAP2": [
                [ "AS1", "AS2", "AS3" ], [ "AS3" ]
              ]
          }
      }
    }
  }
}

--example
Content-Type: application/alto-propmap+json

{
  "property-map": {
    "pid:AS1": { "entry-point": "http://172.25.0.10:8888/escape" },
    "pid:AS2": { "entry-point": "http://172.26.0.10:8888/escape" },
    "pid:AS3": { "entry-point": "http://172.27.0.10:8888/escape" }
  }
}
```

Road Ahead

- Collect WG feedback
- Should the extensions be adopted?
- Define a more elaborated NFFG object to support extended parameters.
  E.g.:
  - Monitoring parameters
  - Resource requirements, etc.
- Present this work in the upcoming IEEE WCNC'18 (Barcelona, Spain)
- Publish the PoC source code in our public repository.

PoC Implementation



Thanks!

- The authors would like to thank the support of Ericsson Research, Brazil.

Backup Slides

Introduction

- 5G network scenarios call for multi-domain orchestration models.
- Multi-provider orchestration operations will require the information exchange across Multi-domain Orchestrators (MdOs).
- Information to be exchanged:
  - Abstract network topology
  - Resource availability (e.g., CPUs, Memory, and Storage)
  - IT Capabilities (e.g., supported network functions).

Multi-domain Orchestration Challenges

- Scalability:
  - Involves the distribution of topology and resource information in a peer-to-peer fashion (MdO-to-MdO). Multi-operator multi-domain environments where the information distribution is advertised in a peer-to-peer model scales linearly.
- Flexibility:
  - Considers that a distributed approach does not allow domains without physical infrastructure to advertise resource capabilities and networking resources. Such procedures consist in deploying and configuring physical peering points for these domains.
- Complexity:
  - Refers to the discovery mechanism to pre-select candidate domains, accounting for resources and capabilities, necessary for an end-to-end network service deployment.