Automating L2 Network Services for Public Cloud Connectivity (AWS, Azure, GCP)

The customer, one of the leading international telecommunication providers, was in need of support for their efforts in automating their network and public cloud accession.

The customer has developed a solution that allows their own clients to connect their sites, located worldwide, to a specific public cloud provider (AWS, Azure, GCP) using a point-to-point Ethernet connection. This Ethernet service is an E-line service where endpoint A is on client's site and endpoint Z is on service provider's router co-located with a public cloud provider interconnection point. What was definitely a drawback of the solution, was that all service fulfillment actions on provider's side were performed manually by their provisioning team which made the whole process of setting up a connection error-prone and lengthy.

What was the challenge?

The challenge was to **support the automation of provider's network and DC infrastructure**. The use case was driven by the commercial launch of a new version of the product for which the end-customer was expected to have a near real-time experience when ordering a new service or modifying an existing one. What was needed in order to achieve it was the lifecycle automation (creating, removing, and updating) of L2 network services. The following cloud providers had to be initially supported:

- **Amazon Web Services** (AWS), Product: Direct Connect
- **Microsoft Azure**, Product: ExpressRoute (via the Equinix ECX Fabric API)
- **Google Cloud Platform** (GCP), Product: Interconnect
- **Internet Exchange** (IX), through IX-API

Additionally, there were several implementation challenges:

- **Variety of APIs to be integrated** – multiple protocols including REST, NETCONF with different design and data representation, service state, and error handling
- **Modeling of complex data structure used by APIs** – e.g. how to describe complex data trees by YANG and used by NETCONF in partner's model
- **Fitting code interacting with APIs into 3rd party orchestrator southbound components** – each API was required to expose CRUD (Create-Read-Update-Delete) interface to manage resources it represented; most

complicated interfaces required development of custom mechanism to calculate the difference between current and expected state

- **Creating top-level service model** – how to reuse multiple resource representations to form one top-level service model, how to handle differences between Cloud Service Providers modules / APIs etc.
- **Testing** – how to ensure that the service provisioned using 3rd party orchestrator will work properly

How we solved it?

In order to meet the requirement of near real-time service lifecycle management, a 3rd party orchestrator was introduced into Service Provider's architecture stack. Amartus team of orchestration experts was responsible for implementation and deployment of modules dedicated to interaction with the leading Public Cloud Providers (AWS, GCP, Azure). Each of these modules consisted of a model part and Python code that was responsible for handling the model lifecycle and performing complementary actions using Cloud APIs.

We also created a generic YANG module responsible for provisioning resources using NETCONF. Based on this generic representation we prepared a data model for each of the network resources required to manage the connectivity. Additionally, Amartus team took part in the main service model implementation and testing (for which dedicated testing framework was developed).

Where's the network innovation?

Amartus involvement and expertise brought added value to the project in terms of:

- **Provisioning of complex network service** in (fully) intent-based manner
- **Management of both CSPs access and E-line service** by one entity and using one model
- **Significantly expediting service delivery time** for major international telecommunication provider