Machine-learning based orchestration of slices (MLO)

1.1. Scope of this document

This document is to describe the purpose of the demonstration and the architecture of the test infrastructure that was used by the NECOS consortium. This test infrastructure is not precluding that any other can be used to run de demonstration. The guide to install the software in the substrate resources is provided in the README file, in the same repository as the software.

1.2. Introduction

In this demonstration, the Tenant hosted at CPqD requests multiple slice allocations to the Slice Provider hosted by UFU that uses the Resources Providers hosted in the same institution as presented in **Figure 1**.

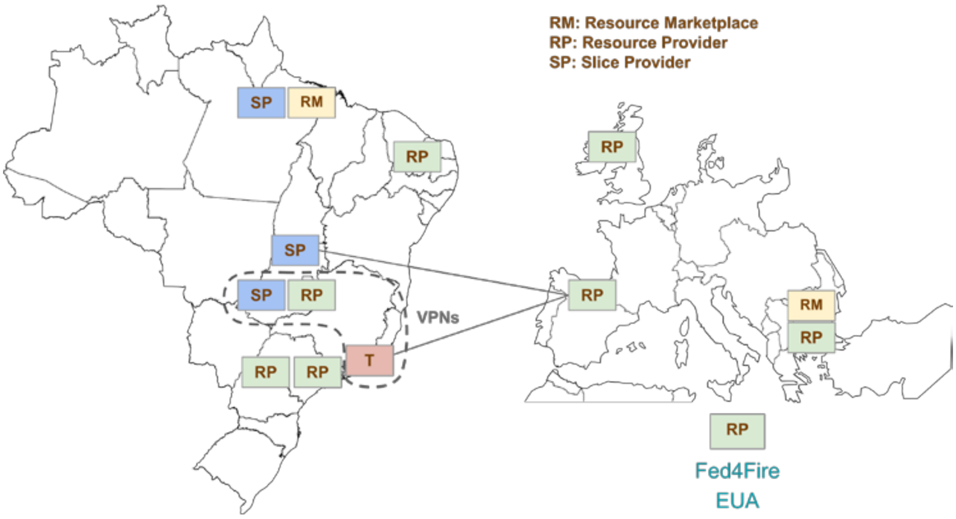

Figure 1. Instantiation of MLO on the experimental infrastructure.

1.3. License

All the source code developed within the NECOS project, and made available as OSS, is released under the Apache License – Version 2.0[1]

1.4. Objectives

The objective of this demo is to show how machine learning algorithms can be useful for the orchestration of slices. Two modules of NECOS Architecture, IMA and SRO, were

[1] http://www.apache.org/licenses/LICENSE-2.0

specifically extended to enable intelligent monitoring and intelligent elasticity orchestration, respectively.

Given the multitude of elements composing slices' infrastructure and the presence of multiple domains in which slice parts are spread to create an end-to-end slice, it is natural to expect considerable overhead to monitor and to move such data from the IMA towards the SRO. Considering such characteristics that can pose a scalability problem, this demo provides intelligence to IMA, enabling it to perform automatic selection of features to be monitored. Such selection is performed on a per-slice basis, requiring that the SRO provides a target KPI to IMA, so it can select a set of features that better describes the behavior of such KPI. We refer to the entire set of infrastructure metrics as full feature set, i.e., the set of metrics regarding the entire infrastructure composing the slice. We refer to the result of the selection mechanism based on machine learning as selected feature set, i.e., a set of K essential metrics selected among all infrastructure metrics composing the slices.

The IMA, in this demo, performs the collection of all infrastructure metrics (full feature set) in longer intervals (less frequently) when compared to the frequency in which the selected feature set is collected. It is expected that along the lifetime of a given slice, the target KPI can evolve and, as a consequence, the selected feature set can probably present a different composition. This effect is justified, for example, by sazonalities related to the usage of the slice. By collecting the full feature set in less frequent intervals, the IMA is capable of automatically updating the selected feature set.

IMA, after the selection of the features, provides the monitoring data to the SRO, according to the time interval specified by IMA. As a result, the intelligent version of the IMA presented in this demo reduces the volume of data being pushed towards the SRO (the selected feature set), not only improving monitoring scalability, but also removing noisy data from the monitoring information. Such noise removal has the potential to improve accuracy in the operations performed by the SRO during slices orchestration.

The intelligent SRO implemented in this demo consumes the selected feature set provided by the intelligent IMA with the goal of supporting elasticity decisions. To do it, the intelligent SRO performs four steps: 1) KPI Estimation; 2) SLA Prediction; 3) Slice Resources Optimization; and 4) Enforcement of Slice Modifications.

The SLA for a given slice, in this demo, is associated to a KPI. Such KPI can be related to the slice infrastructure, considering metrics related to, for example, CPU, memory, network traffic, and others. The most interesting aspect of this demo is that, differently from the other SRO implementation, such KPI does not have to be generic and can also be directly related to a Service KPI. As such in the demo, we actually used Read and Write Response Times of a Cassandra-DHT running as the Service deployed in a given slice. The SRO estimates the current state of such Service KPIs as a way of verifying whether the slice SLA is being violated or not. An example SLA that might be adopted during the demo is to target Read Response Times below 50 milliseconds for verifying the Slice SLA conformance. Besides continuous values, the same solution presented in this demo can be applied for services with discrete classes, like videos in high or low resolution.

The first step taken by the SRO, named as KPI Estimation, consumes the monitoring data provided by the intelligent IMA, feeding it into a supervised machine learning model (regressor/classifier), which is trained to estimate the current state of the target Service KPI. Basically, it is done in order to associate the slice's infrastructure measurements fluctuations with the chosen Service KPI.

By having the history of estimations, the intelligent SRO is capable of performing the second step, named SLA Prediction, which foresees what the state of the SLA will be at a given point in the future. This enables the SRO to proactively tune the slice, preparing it for the condition seen in the future.

The third step, perhaps the most complex out of the four steps listed, has the duty of designing the new slice infrastructure arrangement, capable of handling the condition foreseen by the second step taken by our intelligent SRO. This third step, by itself, opens several research possibilities, including root cause analysis, resource optimization, and others of even higher complexity.

In this demo, we consider the existence of a set of slice flavours. A tenant, when requesting a slice to NECOS – similarly to what is done nowadays when requesting a virtual machine at Amazon – informs the flavour which will be applied to its own slice and, additionally, a set of flavours allowed to accommodate possible SLA fluctuations. The demo detects in step number two whether the SLA is going to be violated or if it is going to be under conformance with a lighter KPI condition, and step number three optimizes the slice flavour, among a set of flavours, which is capable of keeping the SLA under conformance and, at the same time, with moderate resources consumption, i.e., the service will not face SLA violation due to lack of resources, but it will not be running with a set of resources that is not really necessary, given the current service condition estimated by the SRO, as well.

The fourth step does not restrict itself to adjustments in terms of slice infrastructure, already specified in NECOS Architecture, it also encompasses intelligent adaptations. Besides the communication among SRO and DC/WAN Slice Controllers, the intelligent SRO: 1) requests to IMA the update in the composition of the feature set being monitored, ideal to the new slice flavour; 2) updates the trained model being used in step number one, which estimates the Service KPI; and 3) updates the predictive models used in step number two. In short, step number four performs an overall, infrastructure and intelligence, adjustment in this demo.

1.5. Workflow

The demo departs from an established end-to-end slice with a certain service running inside it. As a first step in the demo, the SRO provides IMA with the KPI that it wants to be estimated. As mentioned before, the demo showcases a Cassandra-DHT type of service. It also has a second application, which is a Video-on-demand Service based on Dash Price Chart (DASH).

As seen in **Figure 2**, upon receiving the KPI from SRO (step 1), the intelligent IMA recovers (step 2) from the infrastructure providers the full feature set (i.e., the measurements related to the infrastructure) already monitored by the VIM/WIM. Figure 28 suggests the usage of local databases at infrastructure providers to store monitoring data, but depending on the monitoring technology within infrastructure providers, IMA might assume the responsibility of storing it.

After step 3, with all monitoring data at hand (full feature set), the IMA performs feature selection using machine learning. It selects the features to monitor using the Service KPI as target metric, i.e., it defines the composition of the selected feature set with K essential metrics that better represent the given Service KPI and executes some tasks in parallel as a consequence of step 4. The set of K features and the respective monitoring history is returned to SRO (step 5), so it can train its KPI Estimation module, at the same time that IMA adjusts the monitoring tools deployed within infrastructure providers. As mentioned before, the intelligent IMA deploys two monitoring instances, one responsible for collecting the full feature set (step 6) at longer time intervals (mainly used to support feature selection) and a second instance responsible for delivering live monitoring data corresponding to the selected feature set (step 7). The definition of both collecting intervals can be explicitly specified by the tenant or by analysing learning curves.

Figure 2also suggests a direct communication from VIM/WIM Monitoring module towards the SRO to deliver live monitoring data (step 8). This demo implementation uses a Pub/Sub system, based on Apache Kafka, which removes several push cycles from the overall monitoring system. Basically, we assume that required adaptations in the monitored data is performed locally at the infrastructure providers, for example, by deploying the functionalities of IMA in a distributed manner among the slice parts composing slices. Such design contributes to the deployment of real time orchestration of slices.

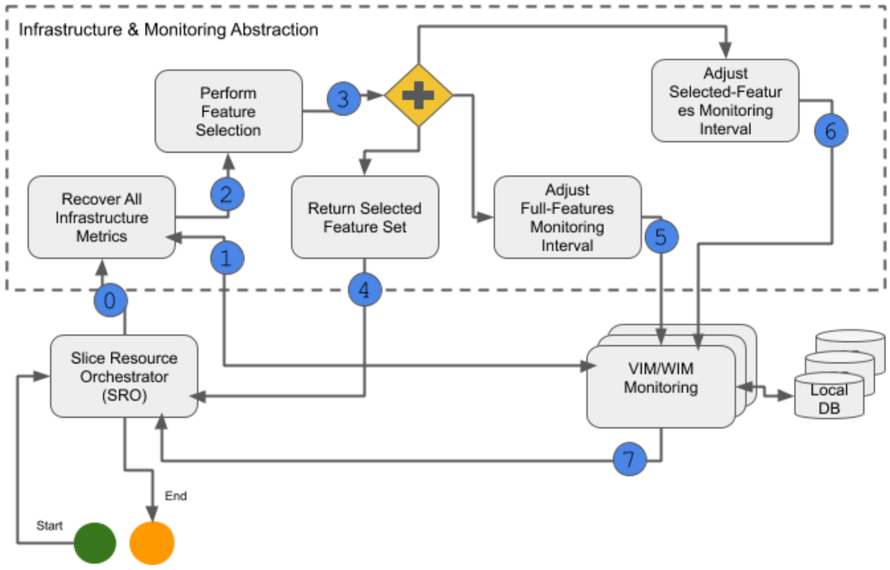
Infrastructure & Monitoring Abstraction
Perform Feature Selection
3
Adjust Selected-Featur es Monitoring Interval
6
2
Recover All Infrastructure Metrics
Return Selected Feature Set
Adjust Full-Features Monitoring Interval
5
1
0
4
Slice Resource Orchestrator (SRO)
VIM/WIM Monitoring
Local DB
7
End
Start

Figure 2. Intelligent IMA workflow for feature selection.

Figure 3 updates the elasticity workflow presented in Deliverable D5.1 of NECOS, presenting the machine learning extensions as green steps. This figure depicts the four steps SRO performs in this demo to support elasticity. This figure also accommodates the proposed communication directly from the VIM/WIM monitoring systems towards the KPI Estimation module of the SRO, as mentioned above.

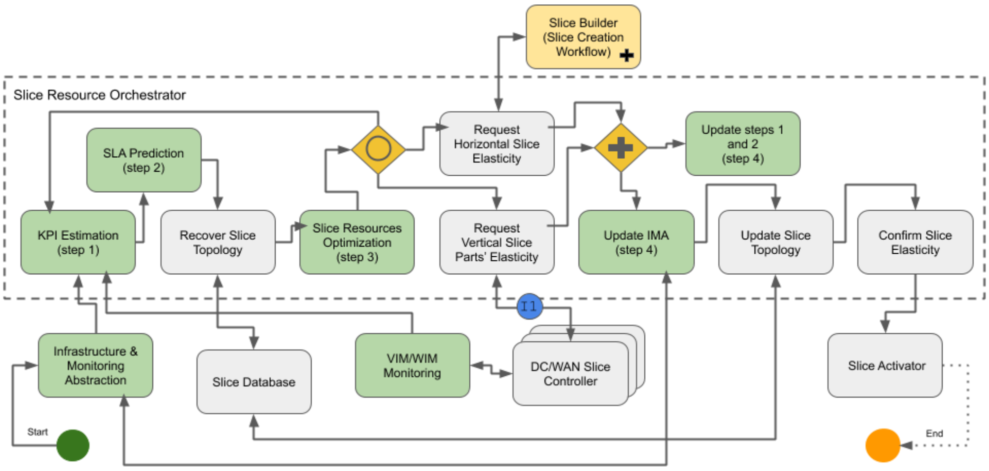
Slice Builder (Slice Creation Workflow)
Slice Resource Orchestrator
SLA Prediction (step 2)
KPI Estimation (step 1)
Recover Slice Topology
Slice Resources Optimization (step 3)
Request Horizontal Slice Elasticity
Request Vertical Slice Parts' Elasticity
Update steps 1 and 2 (step 4)
Update IMA (step 4)
Update Slice Topology
Confirm Slice Elasticity
I1
Infrastructure & Monitoring Abstraction
Slice Database
VIM/WIM Monitoring
DC/WAN Slice Controller
Slice Activator
Start
End

Figure 3. Intelligent SRO workflow for elasticity.

- It is important to highlight the possible outcomes of step 3, the decision can indicate the need for vertical and/or horizontal elasticity, both including upgrade and/or downgrade of infrastructure resources. But, another important outcome of step 3, is to keep the slice in its current form, i.e., returning the loop to KPI Estimation (step 1) of the intelligent SRO. This latter case represents the scenario in which the optimization of slice flavours indicates that current slice arrangement is the best among the available options, for example.