D3.2

Implementation of selected release 1 AI-driven inter-domain network control, management and orchestration innovations

NXW

Funded by the European Union

Revision v0.1

Work package	WP3
Task	T3.1, T3.2, T3.3, T3.4
Due date	31-12-2023
Submission date	31-12-2023
Deliverable lead	NXW
Version	V1.0
Authors	Pietro G. Giardina, Matteo Ravalli, Juan Brenes, Giada Landi (NXW) Sebastian Robitzsch, Chathura Sarathchandra, Renan Krishna (IDE) Salvatore Spadaro, Luis Velasco, Fernando Agraz, Marc Ruiz, Davide Careglio, Albert Pagès (UPC) Alejandro Calvillo, Constantine Ayimba (UC3M) Roya Doostnejad, Rafael Rosales (INT) Péter Szilágyi, Tamás Kárász, Szabolcs Nováczki, Zoltán Vincze, Csaba Vulkán (NOK) Carla Fabiana Chiasserini, Giuseppe Di Giacomo, Somreeta Pramanik (POLITO) José Luis Cárcel, Luis F. González (ATOS)
Reviewers	Otilia Bularca (SIM) Luis Miguel Contreras (TID) Antonio De La Oliva (UC3M)

Abstract

This document starts from AICP functional architecture and defines a methodology to select those functional modules required for the implementation of an AICP prototype, aiming at exposing a minimum set of functionalities (provisioning and decommissioning) for the lifecycle management of E2E Deterministic services. SW design and development plans are reported for the various component of the AICP, along with a general long-term implementation roadmap.

Keywords

Control Plane, Cross-domain, Data Plane, SW architecture, DetNet, 3GPP, Time management

Document revision history

Version	Date	Description of change	Contributor(s)
v0.1	12-09-2023	Initial ToC	Pietro G. Giardina (NXW)
v0.2	06-12-2023	Draft for internal review	Authors in initial table
v1.0	20-12-2023	Submission version	Authors and reviewers in initial table

Disclaimer

Copyright notice

Document information

Nature of the deliverable	OTHER*	
Dissemination level		
PU	Public, fully open. e.g., website	✓
CL	Classified information as referred to in Commission Decision 2001/844/EC	
SEN	Confidential to PREDICT-6G project and Commission Services	

* Deliverable types:

R: document, report (excluding periodic and final reports).

DEM: demonstrator, pilot, prototype, plan designs.

DEC: websites, patent filings, press and media actions, videos, etc.

OTHER: software, technical diagrams, etc.

Table of contents

List of figures

List of tables

Acronyms and definitions

3GPP	Third Generation Partnership Project
AI	Artificial Intelligence
ADRF	Analytics Data Repository Function
AICP	AI-driven Multi-stakeholder Inter-domain Control-Plane
AI/ML	Artificial Intelligence/ Machine Learning
API	Application Programming Interface
BE	Best Effort
CBRM	Constant Bit Rate Model
DetNet	Deterministic Networking
DB	Data Base
DL	Decentralized Learning
DT	Digital Twin / Digital Twinning
E2E	End-to-End
FDS	Federated Domain Set
FL	Federated Learning
FO	Federating Orchestrator
HT	High Throughput
IAM	Identity and Access Management

JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LL	Low Latency
MD	Managed Domain
MDP	Multi-domain Data-Plane
ML	Machine Learning
MS	Management Service
MTLF	Model Training Logical Function
NBI	North Bound Interface
NWDAF	Network Data Analytics Function
PACT	Performance-Aware Compression and Training
PC	Path Computation
PCE	Path Computation Element
POP	Pipeline Orchestration Platform
QoS	Quality of Service
REST	Representational State Transfer
RL	Reinforcement Learning
SBI	South Bound Interface
SLA	Service Level Agreement
SW	Software

TD	Technological Domains
TFS	TensorFlow Serving
TO	Training Orchestrator
TSN	Time Sensitive Networking
TWT	Target Wake Time
UC	Use Case
WP	Work Package
XML	Extensible Markup Language

Table of partners

Short Name	Partner
UC3M	Universidad Carlos III de Madrid
NOK	Nokia Solutions and Networks KFT
ERC	Ericsson Espana SA
INT	Intel Deutschland GMBH
TID	Telefonica Investigacion y Desarrollo SA
ATOS	ATOS IT Solutions and Services Iberia SL
GES	Gestamp Servicios SA
NXW	Nextworks
COG	Cognitive Innovations Private Company
SIM	Software Imagination & Vision SRL
POLITO	Politecnico di Torino
UPC	Universitat Politecnica de Catalunya
CNR	Consiglio Nazionale delle Ricerche
UNIPD	Universita degli Studi di Padova
IDE	Interdigital Europe LTD

1 Executive summary

The document discusses several aspects related to the software implementation of PREDICT-6G's Control plane, the AICP (AI-driven multi-stakeholder Inter-domain Control-Plane). Starting from AICP's functional architecture, a subset of the different Management Services (MS) previously defined have been selected for implementation, with the aim at implementing a prototype able to provide a minimum set of functionalities for the lifecycle management of E2E deterministic services: provisioning and decommissioning. To achieve this objective, a specific development methodology has been defined, taking into account 3 main aspects: the selection of MSs, the technology constraints from the MDP (PREDICT-6G Data Plane), and time available to release the prototype.

The selection process represents the first part of the development methodology and is based on specific criteria that considers two main sources of information: i) the operation workflows, defined together with the AICP's functional architecture, and ii) the set of use cases defined in the project. Additional refinements allowed to select MSs initially not considered by the main selection criteria but crucial for a deterministic service control plane i.e., time synchronisation MSs.

The AICP aims at being as much as possible technology-agnostic. Nevertheless, some modules interacting with the MDP are directly affected by the network technology. For those specific cases, a technology-driven implementation is considered: an AICP SW module may encompass one or more MSs, characterized by different implementation flavours based on the MDP technologies they must support.

The available time up to the AICP prototype release is organized into Development Cycles, lasting 3 months, at the end of which feedback is collected from the implementation activities, being used to i) tune the AICP functional and SW design and ii) organize integration activities, internal to AICP and with MDP modules.

The application of the proposed development methodology led so far to two key results reported in the document:

- **Initial AICP SW design and early implementation.** A set of software modules covering all the required functions to realize the AICP prototype. Each module reports a SW design, a high-level release plan and, where available, an early implementation description.
- **Initial implementation roadmap.** A roadmap that provides the expected advancements per each development cycle up to the AICP prototype.

2 Introduction

Deliverable D3.1 (PREDICT-6G/D3.1/ 2023) presented the AICP functional architecture aiming at design a control-plane based solution for the management and control of E2E Deterministic services across a multi-administrative multi-technology data plane. The result of such a theoretical work is a set of Management Services (MS, i.e., the AICP functionalities) residing in the different Management Domains (MD) characterizing the hierarchical architecture of the AICP. From a practical point of view, the final achievement of the PREDICT-6G control plane implementation path is the release of an AICP prototype that offer the minimum set of functionalities for the lifecycle management of an E2E deterministic service (i.e., at least provisioning and decommissioning) across the various technological domains supported by the project's data plane (MDP). Given such a limited set of service management functionalities, the implication is that not all the MSs defined for the AICP need to be targeted for implementation (or can be partially targeted).

The scope of this document is to report on the SW modules composing the AICP both in terms of design and early implementation, on the methodology used to select their functionalities from the AICP functional architecture, and on the roadmap toward an AICP prototype.

The methodology defined for the selection of the MSs candidate for implementation is described in Section 3 which includes plans how to organize the time for the development of the different SW artifacts and their storage and consideration concerning the different data plane technologies to be supported. Section 4 reports the design of the different SW modules that build the AICP functionalities, complemented with high-level discussion on which tools, frameworks and SW libraries could be exploited for the implementation of the given module. Early implementations are reported when available. The release plans of the different modules are reflected in the long-term roadmap discussed in Section 5.

Finally, the document is complemented with information related to the service information and lifecycle modelling, which are described in dedicated Appendixes, where the operation workflows from D3.1 are also reported as reference for the reader.

3 Methodology

In D3.1, the functional AICP architecture has been presented as a set of Management Services deployed in a hierarchical control plane architecture, as depicted in **Figure 3-1**.

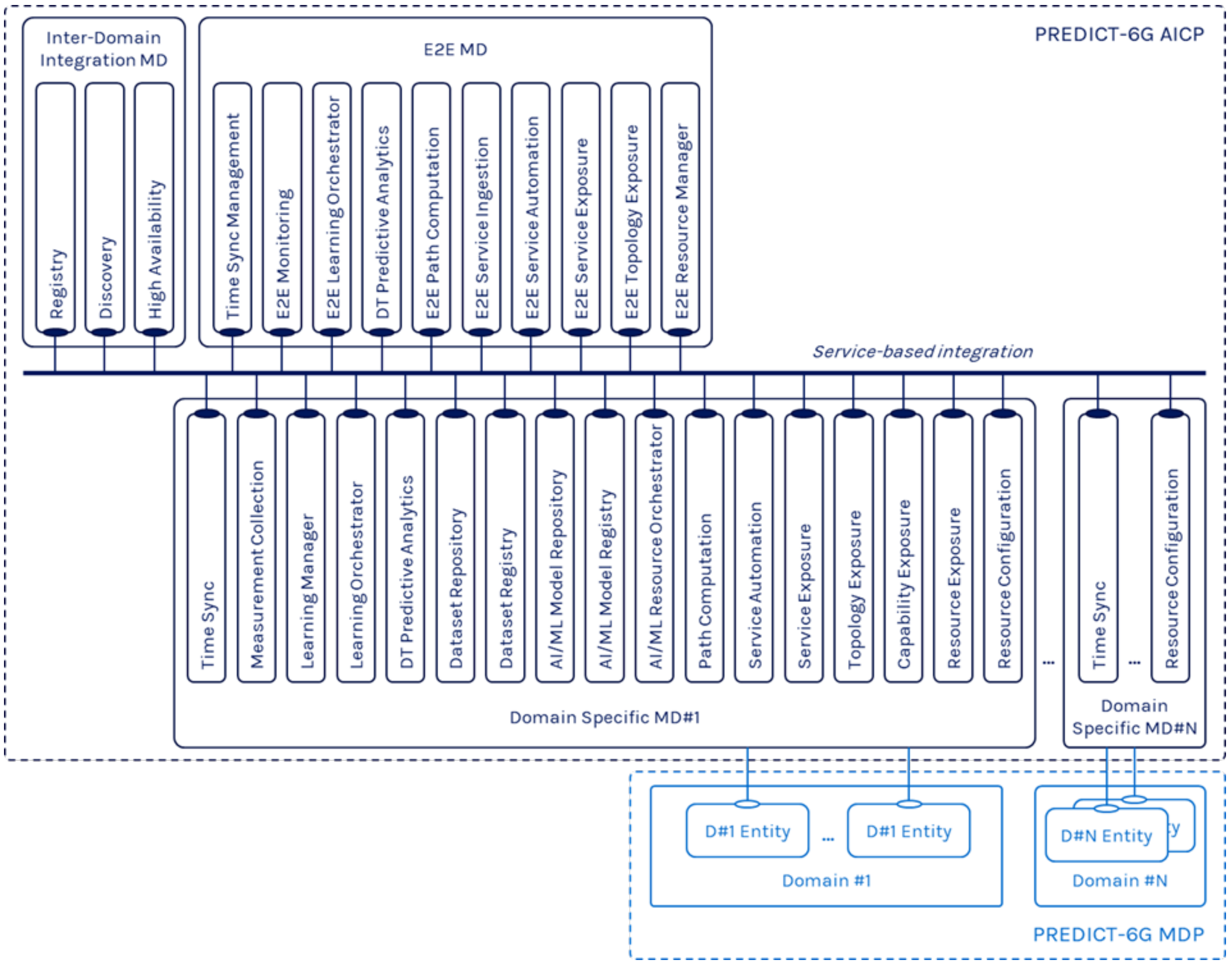


Figure 3-1. PREDICT-6G AICP functional architecture

This section discusses the general methodology used for the selection of the MSs to be implemented for developing the AICP prototype. It is important to highlight that such a selection process is dynamic, and the outcomes may vary as the project moves forward. In particular, the following factors have been identified as the ones that can affect the AICP implementation:

- **Use Cases tuning**. The set of UCs reported in D1.1 (PREDICT-6G/D1.1/4-5, 2023) could be subject to refinements and this may lead to an update of the requirements that need to be fulfilled by the AICP.
- **MDP implementation choices**. The features for the programmability of the data plane provided by each domain and the way they are exposed impact the implementation of those MSs that should exploit them for e.g., topology management, characteristic exposure (services, resources, capabilities).
- **PREDICT-6G E2E Architecture considerations.** AICP is just a part of a project aiming to create and maintain determinism across multiple and heterogenous network domains. This implies that some affecting AICP decision could be taken based on aspects that goes beyond the scope of AICP itself.
- **Feedback from AICP implementation.** As the implementation activities proceed, some MSs or part of them could become obsolete and/or no longer required. This can also imply a refinement of the AICP functional design, to be reported in D3.3.

Handling of those factors requires a meaningful collaboration effort from different work packages in the project that complement and refine both the selection process and implementation approach described in the following sub-sections.

3.1 Management Services Selection Process

The MSs selection process relies on 2 main sources of information:

1. Operational workflows reported in D3.1 (PREDICT-6G/D3.1/9, 2023)
2. Use Cases, as defined in D1.1

The source (1) provides an overview of the interactions between the different MSs i.e., who interacts with whom. In addition, such workflows provide hints on the minimum set of functionalities, exposed by each MS, that should be implemented to build the workflows themselves. In this process the workflows considered are related to the E2E deterministic service provisioning and decommissioning, i.e., the minimum set of operations required for the lifecycle management of any service. Both workflows and related sequence diagrams are reported in Appendix, Section 8.1.

The Use Cases, source (2), determine the technological domain to be targeted thus directly affecting the implementation of the MSs resides in technology specific MDs in terms of interfaces to be implemented, KPIs to be monitored, etc.

The combination of these 2 sources of information leads to the selection of a minimum set of MSs and related functionalities to be implemented to build the prototype of AICP.

3.2 Module implementation flavours and common ground

The implementation of MSs directly interacting with the data plane would require supporting multiple network technologies, due to the heterogeneous nature of the MDP. To simplify their design and related implementation, one idea is to create technology-driven implementation of a certain MS (or group of MSs), e.g., a Resource Exposure MS for DetNet and one dedicated to 3GPP, same for Topology Exposure, Resource Configuration, etc. This requires that the implementation of the AICP must be flexible enough to adapt to any change of the MDP, by offering the possibility of dynamically plugging in the specific technology software modules. From a practical point of view, an implementation approach based on microservice paradigm can provide that degree of flexibility, extensible to all the targeted MDs, including the E2E one.

Such dynamic plug-in of different SW module would be assisted, by design, by a set of MSs dedicated to the management of the other MSs designed for the control of the deterministic services. Such MSs reside in a dedicated MD, called Inter-domain Integration MD (**Figure 3-1**), whose role is detailed in dedicated section of deliverable D1.2 (PREDICT-6G/D1.2/5.3.3, 2023). Since the Inter-domain Integration MD does not play a direct role in the control and management of deterministic service, its implementation is considered optional and is not discussed in this document.

In terms of SW artifact management, WP3 exploits the GitLab facility provided by UC3M that offers a dedicated space for PREDICT-6G. All the open-source components of the control plane will be stored in specific repositories under the AICP group, available at the following URL:

https://gitlab.netcom.it.uc3m.es/predict-6g/aicp

Modules whose source code is not open and/or public will be stored in repositories maintained by the respective owners.

3.3 Development Time Management

It is important to note that the implementation of a prototype for the AICP must be considered in the long term and will be reported in D3.4 at M27. In this regard, a roadmap is reported in Section 5 while this section explains how such a roadmap has been derived.

The base idea is to organize the development activities in cycles lasting 3 months, in a "loosely agile" manner. The release of the modules building the AICP is aligned with the end of a development cycle with the aim to realize an integrated prototype at the end of the last cycle. Each development cycle consists of 2 timeframe called Implementation Time and Synchronisation Time, as shown in **Figure 3-2**.

Funded by the European Union

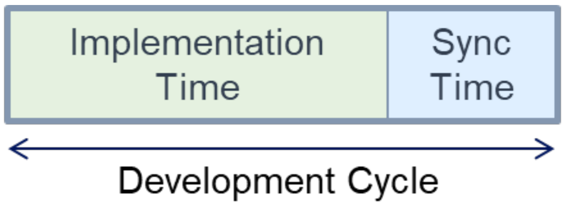


Figure 3-2. Development Cycle composition

The Implementation Time is dedicated to design, coding and testing activities of the software modules. Owner of multiple modules can also perform integration activities where possible. The Synchronization Time is dedicated to test and integration activities between modules belonging to different owners (intra-AICP integration) and with the functionalities exposed by the MDP for the different data plane technologies. The duration of this time interval is 2-4 weeks that may increase (as shown in **Figure 3-3**) since as the project proceed, it is expected to have more integration than implementation activities.

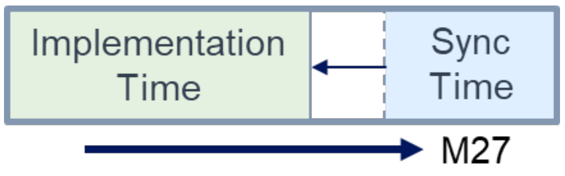


Figure 3-3. Development Cycle composition variation

At the end of Synchronization Time, the Development Cycle terminates, an AICP software release is provided and the feedback for enhance the function design of the control plane is collected.

As depicted in **Figure 3-4**, given the duration of the cycle, a total of 4 Development Cycles is considered form M13 to M24. The period in M10-M12 is dedicated to the early implementation activities, reported in this deliverable, while in M25-M27 it is expected the release of first integrated AICP prototype.

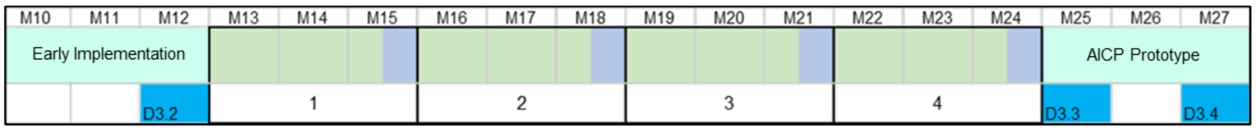


Figure 3-4. Development Cycles in M10-M27

Section 5 describes a more detailed version of the implementation roadmap, providing the expected advancements per each SW module at the end of which development cycle.

4 Development of AICP

Following the selection criteria described in Section 3.1, it can be noted that 3 MSs are not explicitly involved in any operational workflow defined for the AICP:

- Time Sync
- Time Sync Management
- E2E Resource Management

Proceeding with a more detailed analysis, it can be noted that the only MS that can be considered not mandatory for implementing the minimum set of service lifecycle functionalities is the E2E Resource Management MS. It is worth noting, indeed, that the time synchronisation is a key concept for managing time-sensitive services and achieving determinism while the functionalities exposed by the E2E Resource Management are not crucial for the AICP and, when required, can be covered by some other MSs. For that reason, the E2E Resource Management can be considered a complementary element of AICP functional architecture and not targeted for the implementation. This does not prevent that feedback collected during the implementation cycles and future refinements of the operation workflows will involve the E2E Resource Management or part of its own functionalities, making its implementation necessary for AICP objectives.

In **Table 4-1** is reported the list of MSs selected for the implementation, aggregated by Development Area.

Table 4-1. List of MSs selected for AIPC implementation

Development Area	**Description**	**AICP Coverage (MSs)**
Time Sync	Time synchronisation service across the whole AICP infrastructure.	• Time Sync • Time Sync Management
Service Ingestion	Deterministic service request parser and requestor privileges checker	• E2E Service Ingestion
Exposure Services	Exposure of information set (capability, resources, etc.) from TDs to MDs and intra-AICP i.e., from MDs to E2E MD.	• Topology Exposure • Capability Exposure • Resource Exposure

		• E2E Topology Exposure
Service Automation	Ensures the correct deployment and maintenance of deterministic E2E services across multiple heterogeneous infrastructures. Exposure of information related to E2E and MD deterministic services	• Service Automation • E2E Service Automation • Service Exposure • E2E Service Exposure
Path Computation	Computes best paths intra and inter-domain for deterministic services	• Path Computation • E2E Path Computation
Digital Twin	Network modelling for KPI estimation and synthetic data generation	• DT Predictive Analytics • E2E DT Predictive Analytics
Resource Configuration	Configuration enforcements at TDs for deterministic service provisioning	• Resource Configuration
Data Collection and Management	Deterministic Service KPI monitoring service at both MD and E2E MD level	• Measurement Collection • E2E Monitoring
AI/ML Algorithmic Frameworks	Centralized/distributed execution of AI/ML tasks	• Learning Orchestrator • E2E Learning Orchestrator
ML Architectural Framework and Interfaces	Manage the lifecycle of AI/ML models, from design and training to deployment in production environments.	• AI/ML Model Registry • Dataset Registry • Model Repository • Dataset Repository • Resource Orchestrator

Funded by the European Union

		• Learning Manager

Figure 4-1 shows the how the implementations cover (in green) the AICP MSs in the functional architecture.

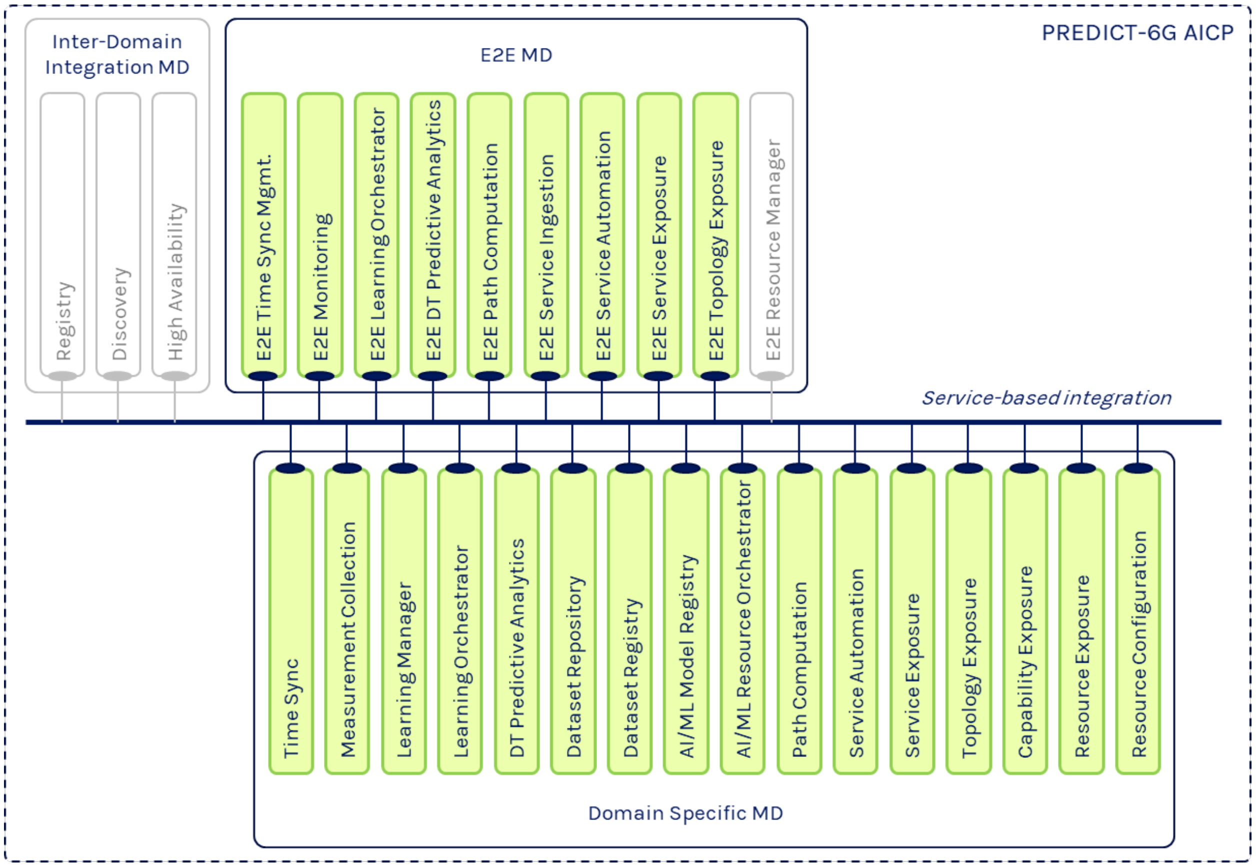


Figure 4-1. AICP MSs implementation coverage (green MSs)

4.1 Time Sync

The role of the time sync module is to set up and maintain consistent time synchronization in the P6G system spanning over multiple technology domains that is crucial for the deterministic operation. To achieve this, a two-level time synchronization management service was defined in D3.1 (PREDICT-6G/D3.1/7.2.1.1-7.2.2.1, 2023) consisting of E2E Time Sync

Management Service and technology domain specific Time Sync Management Service. The design and implementation of the time sync component follows this principle.

4.1.1 SW Design

The Time Sync software component consists of two modules as shown in **Figure 4-2**

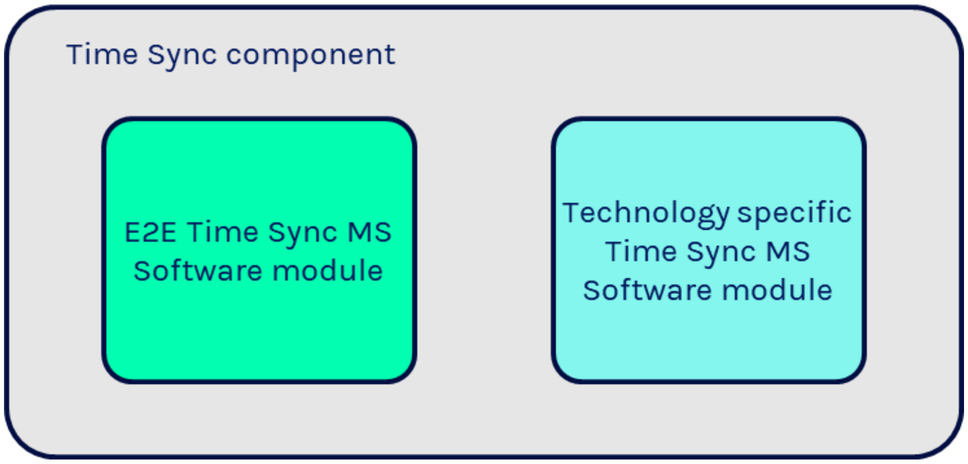

Time Sync component
E2E Time Sync MS Software module
Technology specific Time Sync MS Software module

Figure 4-2. Time Sync software modules

The three main tasks of the E2E Time sync MS are as follows:

- Collection of per domain time sync capabilities: the E2E Time sync MS shall be able to communicate with each domain of the P6G network to collect the time sync capabilities of the respective domains. The E2E Time Sync MS performs this by querying the domain's TS MS and processing the information provided by the domain TS MS.
- Decide and configure GM/Leader and Follower roles: based on the information collected from the domain TS MSs, the E2E TS MS carries out an analysis to determine the best configuration for the Grand Master Leader and Follower roles. After the analysis is finished and the configuration is determined, the E2E MS TS configures the Leader/Follower roles in the domains via sending the necessary configuration to the TS MS of the domains.
- Monitor and assure per domain and E2E time sync state: After the configuration is finished, the E2E TS MS must monitor the operation of the time sync service in the P6G system. To achieve this, the E2E TS MS either periodically queries the TS status from the domain TS MS or subscribes for getting notification from the domain TS MS about TS status changes. The E2E TS MS continuously evaluates the status changes and once

it determines that the TS service's quality is not appropriate it re-evaluates the TS service configuration and determines the right configuration matching the requirements. In the next step, it carries out the necessary configuration changes via the domain TS MS.

The technology domain specific time sync MSs' tasks are as follows:

- Upon the request of the E2E Time Sync MS collect time sync capabilities in the respective domain: once the domain TS MS receives a query from the E2E TS MS regarding the TS capabilities of the technology domain, the TS MS collects the required information via the interface provided by the technology domain and sends the collected information to the E2E TS MS.
- Based on the configuration received from the E2E Time Sync MS, configure the required TS settings in the technology domain: the TS MS provides interface for the E2E TS MS to send time sync configuration regarding the Leader/Follower roles of the domain. Once the TS MS receives a configuration command from the E2E TS MS, it sends the relevant configuration commands to the technology domain via the domain's relevant interface.
- Continuously monitor and report time synchronization status in the respective technology domain: the TS MS shall continuously monitor the status of the time synchronization in the technology domain. TS MS performs this via the methods provided by the technology domain, i.e., either periodically querying the technology domain for the time sync status and/or subscribing for time sync status updates in the technology domain. The status update/changes in configuration shall be reported to the E2E TS MS immediately by the TS MS.

The interaction between the time synchronization MS modules implements the time sync system procedure defined in D1.2 (PREDICT-6G/D1.2/5.3.3.1, 2023) and are shown in **Figure 4-3**.

Funded by the European Union

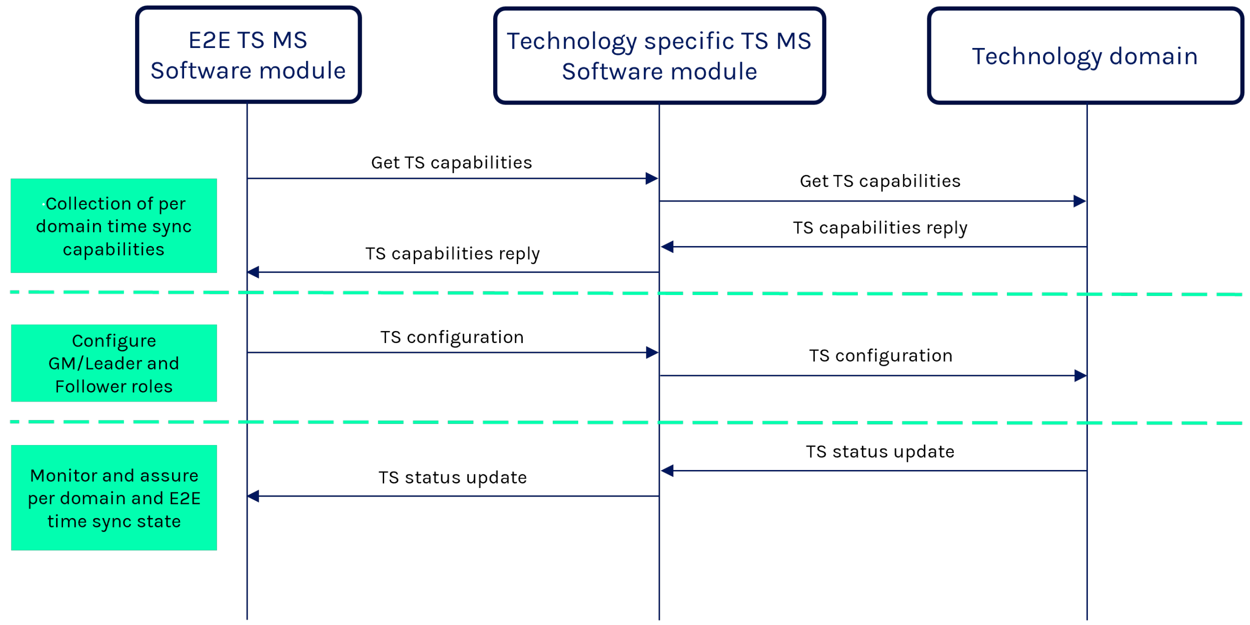

E2E TS MS Software module
Technology specific TS MS Software module
Technology domain
Collection of per domain time sync capabilities
Get TS capabilities
Get TS capabilities
TS capabilities reply
TS capabilities reply
Configure GM/Leader and Follower roles
TS configuration
TS configuration
Monitor and assure per domain and E2E time sync state
TS status update
TS status update

Figure 4-3. interaction of the time sync modules

4.1.2 Early Implementation and Release plan

Currently there is no implementation of the Time Synch Service module available, the implementation work is planned according to the roadmap reported in Section 5. The implementation is planned to be done using Python and OpenJDK.

4.2 Service Ingestions

The Service Ingestion module is in charge of handing lifecycle management requests for E2E deterministic services (Provisioning, Decommissioning) and represent the E2E service management interfaces for AICP external users. With respect to the E2E lifecycle model, reported in Appendix (Section 8.2), the modules perform the operations required when the E2E deterministic service is in the first administrative state: Ingestion.

According with D3.1 (PREDICT-6G/D3.1/7.2.2.2, 2023), the description of the corresponded MS provides a single functionality called "Validate service requests" that implied several sub-functionalities:

1. Request syntax check
2. Requestor privileges check
3. E2E Resource availability check
4. Domain selections

Nevertheless, (3) and (4) are no longer required since (3) is not used (see explanation at the beginning of Section 4) and (4) is performed at the level of E2E path computation by the specific PCE implementation (see Section 4.5). With respect to (2), the check of the privileges will happen with the support of an external centralised Identity and Access Management (IAM) Platform e.g., Keycloak[1], since it can be considered a functionality that involves multiple elements in the AICP, such as the administrative operation, access to the data e.g., monitoring, service status, domains and resources involved, etc.

Without the Service Ingestion module, it would be impossible for a requestor to provision an E2E deterministic service in a centralized manner. This implies that the Service Ingestion is a core AICP module, involved in all the UCs aiming at demonstrating E2E deterministic services.

4.2.1 SW Design

The module exists at the E2E MD and exposes an interface for requestion and terminating E2E deterministic service. As depicted in **Figure 4-4**, the initial SW design considers a REST-based North-Bound interface (NBI).

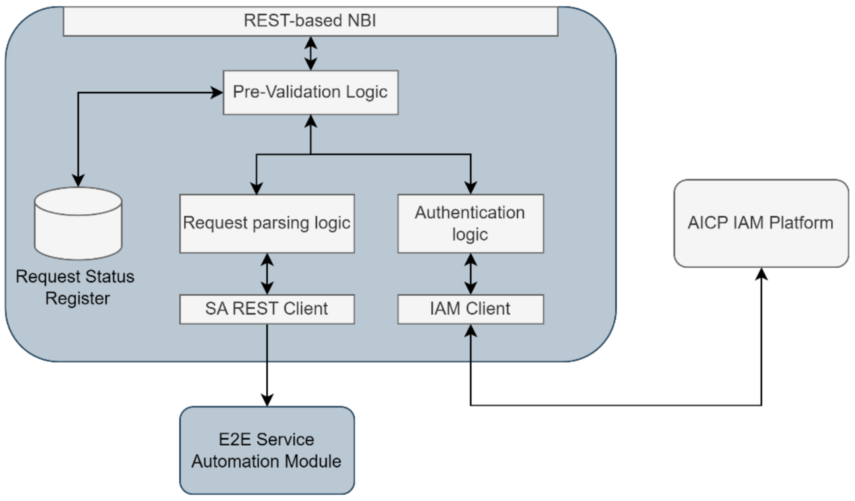


Figure 4-4. Service Ingestion module SW design

[1] https://www.keycloak.org/

The other internal components are:

- **Pre-Validation logic (or Request Pre-validator).** Analise the requests against the NBI and split them in two sub-requests: one directed to the Authentication Logic, to validate the requestor privileges and one to the Service Information parsing logic. It is also in charge of assembling messages with the result of the request that is notified to the requestor and of store this results in specific internal DB, the Request Status Register
- **Authentication Logic.** Verifies the credential and the level of privileges of the user. As discussed above, this task is fulfilled with the support of an IAM properly configured to provide this kind of service in a centralized manner.
- **Request Parsing Logic.** It is in charge of validating the syntax and the form of the request part related to the service. In the case of decommissioning, this validation is minimal since, in that case, the only parameter requested is the E2E service identifier. In the case of provisioning request, the logic must analyze a more complex data structure (JSON or XML) representing the model of the E2E service. This model is defined by using the YANG language and is available in Appendix C, Section 8.3.1.1.
- **Request Status Register.** Maintains the status of each service request in the format:

  ```
  <Request_ID>, <Request_type>, <Requestor_ID>, <Status>, <Information>
  ```

  Where:

  - *Request_ID/Requestor_ID* are the unique identifiers of request and requestor respectively;
  - *Request_type* can be PROVISION or DECOMMISSION. Other type can be added in as the AICP evolves;
  - *Status* is the status of the request: Accepted, In_Progress, Failed
  - *Information* is a field that can contain the reasons in the case of failing “no e2e path found” and/or the ID of the E2E service in the case of success.
- **Clients.** Build the South-Bound interfaces (SBI) of the Service Ingestion module. The IAM client is specific for the platform while the Service Automation client is REST, according to the E2E Service Automation design.

With respect to the NBI, **Table 4-2** reports the initial set of APIs to be exposed.

Funded by the European Union

Table 4-2. List of APIs exposed by the Service Ingestion NBI

Name	Description	HTTP Method	Input/Output
Service Provision	Request the provision of an E2E service with certain characteristics	POST	I: Service description according to the model reported in Appendix C, Section 8.3.1.1 O: A body that follows the model of the items stored in the Request Status Register. A proper HTTP code is also returned e.g., 201 CREATED, 401 UNATHORIZED, etc.
Service Decommission	Request the decommissioning of an E2E service	DELETE	I: ID of the E2E Service O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
Request Status	Request the status of a given requests	GET	I: Request ID (Optional). O: An entry in the format described for the Request Status Register. If the Requets_ID is not provided, all the entries in the registry are returned. Proper HTTP codes are also returned: 200 OK, 404 NOT_FOUND, etc.

4.2.2 Early Implementation and Release Plan

Currently there is no implementation of the Service Ingestion module available, nevertheless, a repository in the project's GitLab is already available:

https://gitlab.netcom.it.uc3m.es/predict-6g/aicp/service-ingestion

The plan is to have this module stable in two releases at M18 and M24, according to the roadmap reported in Section 5. Progressively covering the "Validate service requests" functionality exposed by the corresponded MS in deliverable D3.1.

With respect to the code, tool and framework that can be employed, the idea is to use Python as main programming language. FastAPI[2] is targeted as framework for the implementation of

[2] https://fastapi.tiangolo.com/

the NBI, whose API will be also defined by following the OpenAPI[3] specification. As mentioned, Keycloak is a potential candidate as IAM platform: Python modules already exist and can be used to implement the IAM Client. The Request Status Register can be implemented with a timeseries DB, due to its structure.

4.3 Exposure Services – Topology, Capabilities and Resources

From a general point of view, an Exposure Service can be described as an entity that collects information codified with a given information model and translates them by using another information model, as shown in **Figure 4-5**. It is worth noting that the relationship between the input and output models is not necessarily 1:1 but could be N:1, N:N, N:M, 1:N, etc.

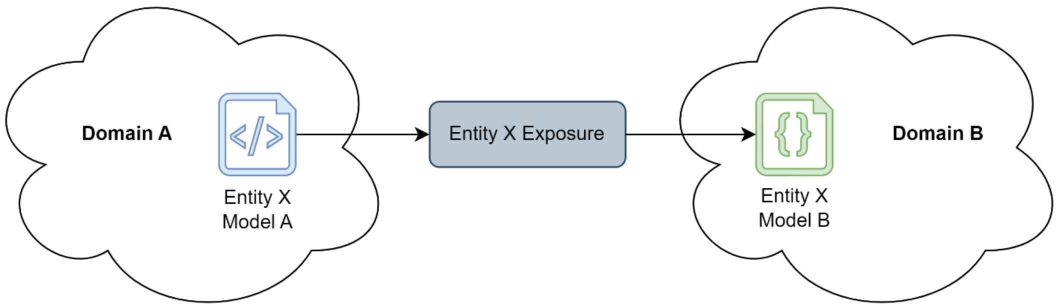


Figure 4-5. Exposure service general behaviour

The AICP needs to know the characteristics of each technological domain to properly enforce configurations and enable the management of E2E deterministic services. Each technological domain exposes multiple set of information concerning its own network e.g., provisioned services, network topology and capabilities, resources status, etc., whose modelling depends on the technology itself. To consume such information, the AICP functional architecture considers several Exposure MSs, in charge of exposing information form a technological domain to the AICP at MD and E2E MD levels. This is shown in **Figure 4-6**, where an exposure chain between domains translates the Entity X (e.g., network topology) information from a Technological Domain specific model to the AICP E2E model. This behaviour is in line with the functionalities reported in D3.1 (PREDICT-6G/D3.1/7.2.1.3-7.2.1.5-7.2.1.6-7.2.1.7-7.2.2.6-7.2.2.7, 2023)

[3] https://www.openapis.org/

for all the exposure MSs: i) obtain information, ii) abstract information (i.e., translate to another information model), iii) expose information.

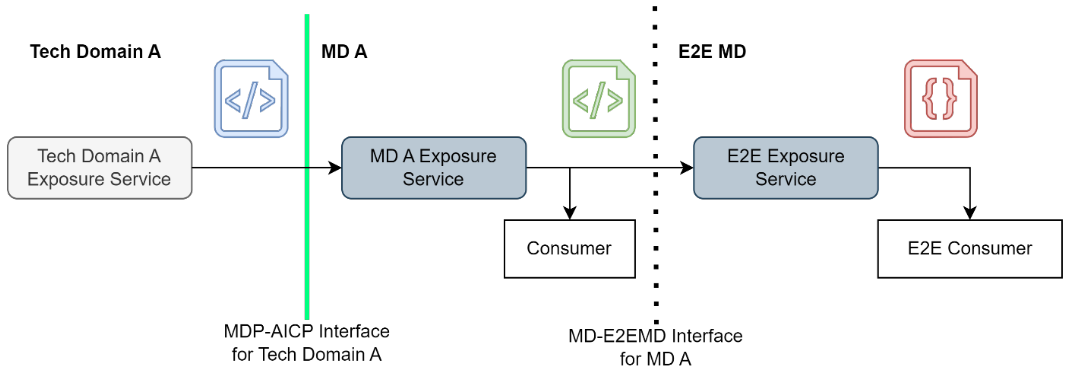

Tech Domain A
MD A
E2E MD
Tech Domain A Exposure Service
MD A Exposure Service
E2E Exposure Service
Consumer
E2E Consumer
MDP-AICP Interface for Tech Domain A
MD-E2EMD Interface for MD A

Figure 4-6. MDP-AICP information exposure chain

4.3.1 SW Design

The development of an Exposure Service is characterized by a model-driven implementation as the internal logic and the interfaces exposed are highly dependent on the models supported. This implies that the information models are crucial for characterization of the module, while the software architecture can be generalized and applicable to all types of Exposure Service, careless of the information exposed. A further element that can affect the implementation details is the dynamicity of the information i.e., the frequency the source information changes: the Exposure Service must be able to timely detect such changes to provide always updated information to the consumers.

A general software architecture for an exposure service is reported in **Figure 4-7**.

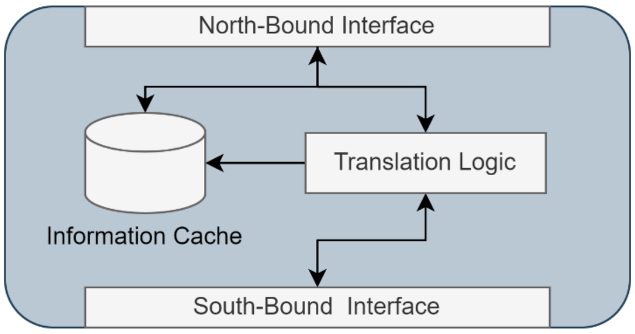


Figure 4-7. General Exposure Service SW architecture

It consists of 4 main elements:

- **North/South-Bound interfaces (NBI/SBI).** They are the Output and Input interfaces, respectively for the translation process. The Exposure Service collects information for a given element from the SBI and makes it available with the new format from the NBI. These interfaces must consider the information dynamicity aspects discussed above by implementing specific communication mechanisms (e.g., request/response, subscribe/notification, etc.)
- **Translation logic.** Provides the logic for translating from a model to another one. Modules such as the Digital Twins may require accessing data in the original format by requesting them through a specific API in the NBI. In this case, the translation logic simply forwards the data from the SBI to the NBI.
- **Information Cache.** Is the internal repository that maintains the data in the output format. Refresh updates depend on both the information dynamicity and the communication mechanism implemented to timely meet the change in the input domain.

4.3.1.1 Data types and domains' considerations

While the SW architecture in **Figure 4-7** is valid for all the Exposure Services, the implementation details depend on the type of the collected information, from which domain is collected, and which domain is the target.

The exposure of the network topology is crucial for several services as topology information not only offers an overview of the network but also includes additional information such as resources available and network capabilities. For this reason, the Network Topology Exposure service incorporates in a single component 3 different Exposure MSs reported in D3.1: Topology, Resource and Capabilities.

The topology models for AICP internal purposes are different at the E2E MD and MDs. In particular, at the MD level, the representation of the topology is uniform i.e., the same per each MD regardless of the technological domain managed but still with the granularity of the network exposed by the MDP i.e., a node per network device. This allows the Path Computation service (see Section 4.5) to compute best paths inside the technological domain. Particularly, the abstracted topology contains a high-level view of the KPIs of the links, namely, range of achievable latencies, available/remaining link capacity and reliability. This, coupled with the identifiers of the nodes, as well as the fields stating the endpoints of the links, the Path Computation can execute weighted graph traversal algorithms to determine the path within the domain as a sequence of physical links.

At the E2E level, the topology graph is more abstract, and the granularity is one node per technological domain, with edge nodes of that domain as ingress/egress points. Thus, a link in the E2E abstracted topology not only represents the inter-domain link but also which are the border nodes in the domain sequence. In addition to that, the nodes at the end-to-end topology representation, besides the domain identifier, also contain a field indicating the technological capabilities of the domain, that is, which is the specific technology (3GPP, DetNet, etc.) that the data plane representation of the characteristics of the inter-domain links is the same as in the internal domain topology model. Having said that, **Figure 4-8** depicts a representation of the two abstraction procedures. An example of the structure of the model and considered fields can be found at Appendix C.

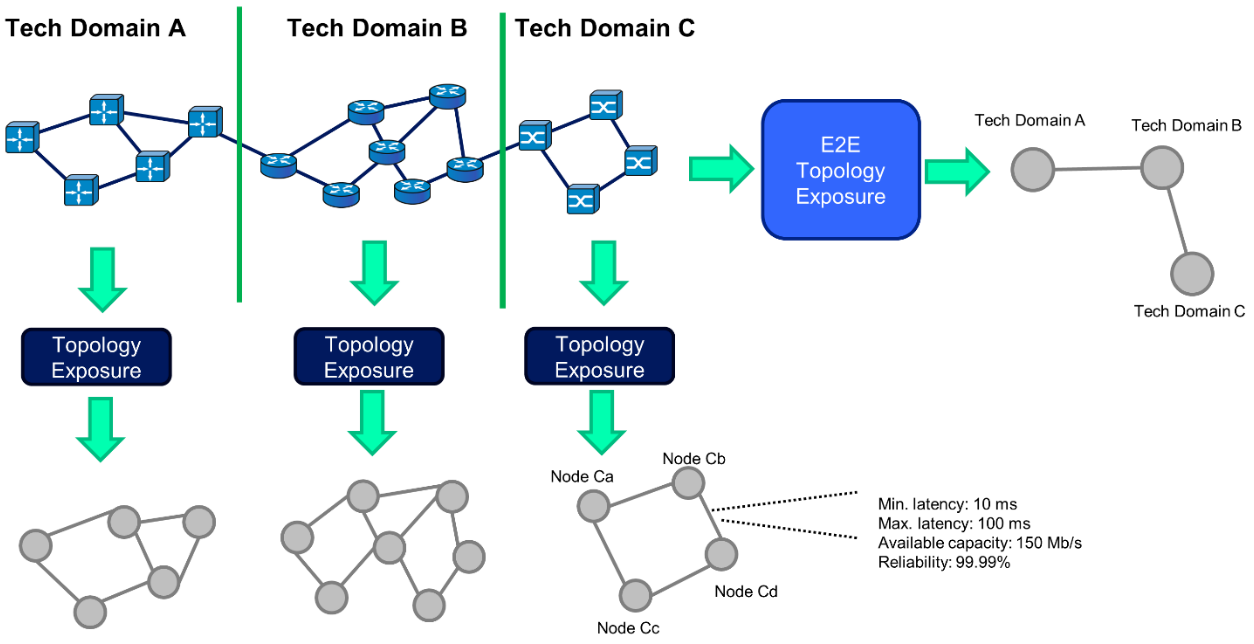


Figure 4-8. Example of abstract topology exposure at end-to-end and domain level

Network topology is subject to changes during the network lifetime with an unpredictable frequency and it is necessary for the Network Topology Exposure services (both E2E and not) to implement a mechanism to refresh their information in near real-time. This mechanism is dependent on the MDP choices, i.e., whether MDP will offer a topology notification services or not. In the first case, the SBI of the Network Topology Exposure service can implement a subscribing mechanism to listen to MDP topology notification. This mechanism can be in turn implemented in the E2E version of the exposure service and listen to changes at MD level. In the case the notification service would not be available, one solution would be to refresh topology at each topology request e.g., at service provisioning time: the related workflow is simple to realize but may take some time to be performed, enlarging the time to fulfil the request. **Figure 4-9** shows both mechanisms.

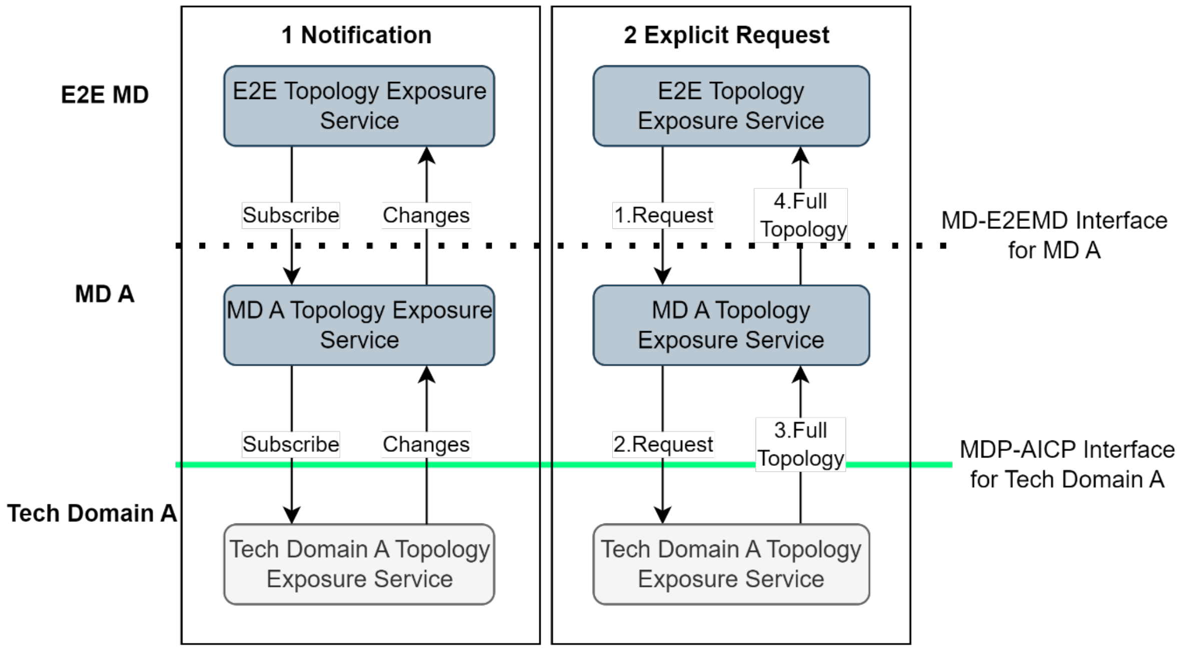

1 Notification
2 Explicit Request
E2E MD
MD A
Tech Domain A
E2E Topology Exposure Service
MD A Topology Exposure Service
Tech Domain A Topology Exposure Service
Subscribe
Changes
1.Request
4.Full Topology
2.Request
3.Full Topology
MD-E2EMD Interface for MD A
MDP-AICP Interface for Tech Domain A

Figure 4-9. Topology information updates 1) Notification based and 2) Request based

The list of APIs exposed by the Network Topology Exposure service is reported in **Table 4-3**.

Table 4-3. List of APIs exposed by both Network Topology Exposure Services' NBI (MDs and E2E MD)

Name	Description	Type	Input/Output
Get Topology	Request information about the full topology as list of nodes and links	HTTP/REST, RPC, Event, etc.	I: N/A O: A body that follows the topology model, depending on the domain the service resides (MD or E2E MD). The E2E topology model is available in Appendix, Section 8.3.2.

Get Nodes	Request information about one or more nodes in the target topology	HTTP/REST, RPC, Event, etc.	I: ID of the node (optional) O: A list with one or more nodes and their characteristics (if the node ID is not provided, all the nodes are returned)
Get Links	Request information about one or more links in the target topology	HTTP/REST, RPC, Event, etc.	I: ID of the link (optional) O: A list with one or more links and their characteristics (if the link ID is not provided, all the links are returned)

4.3.2 Early Implementation and Release plan

Currently there is no implementation for Network Topology Exposure service, nevertheless, a dedicated code repository in the project's GitLab is already available:

https://gitlab.netcom.it.uc3m.es/predict-6g/aicp/network-topology-exposure

The implementation plan is reflected in the roadmap in Section 5 and aim at progressively covering the 3 functionalities reported in D3.1 for the Exposure MSs (i.e, i) obtain information, ii) abstract information, iii) expose information).

- **MD level implementation.** For the Network Topology exists a direct dependency on MDP implementation, so it is expected to have a complete implementation by M24. Anyway, it is planned to build a mock implementation with the NBI that provides topology samples by M15 to proceed with the implementation of the E2E counterpart that will consumes those samples.
- **E2E MD level implementation.** For topology it is expected a stable implementation by M21 to be refined up to M27.

The source code will be produced in Python by also exploiting specific packages for generating codes starting from YANG models, e.g., pyangbind[4]. For the REST interfaces, OpenAPI paradigm will be followed for API specification while FastAPI is the framework targeted for the implementation.

[4] https://pypi.org/project/pyangbind/

4.4 Service Automation

The Service Automation module is the AICP component designed to ensure the correct deployment and maintenance of deterministic E2E services across multiple heterogeneous infrastructures.

The module is composed of two specific Management Services (MS): the E2E Service Automation MS and the Management Domain (MD) Service Automation MS. Consequently, the purpose of the Service Automation module is twofold. On the one hand, the E2E Service Automation MS oversees the proper implementation and management of an E2E service across all technology domains, solving all possible conflicts between each other. On the other hand, the MD Service Automation MS interacts with each technology domain to ensure the closed-loop automation of an accepted service. Further information of both MS and their functionalities can be seen in section 7.2 of D3.1 (PREDICT-6G/D3.1/7.2, 2023).

To ensure closed-loop automation and the correct implementation of a service, the Service Automation module manages the following lifecycle stages:

- **Service provisioning:** Implements the initial configuration of a technology domain to meet service requirements.
- **Service assurance:** Continuous control-loop configuration and monitoring to ensure service lifecycle.
- **Service termination:** Ensures service removal from a technology domain after a service lifetime expiration or after receiving a termination request.

Furthermore, this module will also include another two specific MSs: E2E Service Exposure and MD service exposure. This module will be also responsible for exposing service details to other MSs and/or users requesting specific details of both E2E services and local services in a specific TD.

Regarding its SW development, the Service Automation module will be implemented in all project use cases: smart manufacturing, deterministic services for critical communications and multi-domain deterministic communication. Further information of these use cases can be seen in D1.1.

4.4.1 SW Design

Based on the workflows described in section 9 of D3.1, a Service Automation SW module will be implemented to perform service provisioning and service decommissioning at both E2E and MD levels. Considering the theoretical design of Service Automation Management services, the Service Automation SW module will consist of two submodules: one designed to cover E2E functionalities and the other for addressing MD needs.

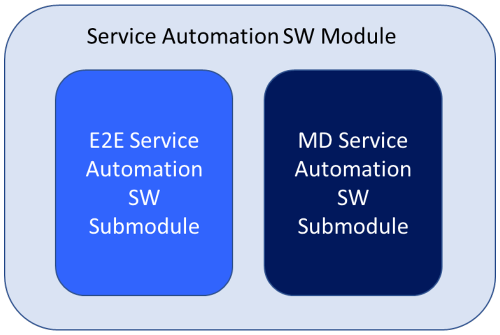

Service Automation SW Module
E2E Service Automation SW Submodule
MD Service Automation SW Submodule

Figure 4-10. Service Automation SW Module and Submodules

Based on the proposed design, this subsection describes the Service Automation SW implementation, considering Service Automation (E2E and MD) interactions with the remaining modules present in the AICP. In this regard, the following subsections provide details on how each Service Automation submodule interacts with the rest of modules, which information is sent from/to each module, technical details of their inputs and outputs, and the interfaces that will be utilised to support each interaction.

4.4.1.1 E2E Service Automation SW Submodule

The E2E Service Automation SW Submodule will cover the functionalities and interactions required to perform E2E Service Provisioning and Decommissioning.

E2E Service Provisioning

Figure 4-11 depicts the service provisioning workflow from the perspective of the E2E Service Automation submodule. This figure shows the implementation details of all the interactions between the modules involved in the configuration and setup of a new E2E service. All service interactions, unless it is specified otherwise, will follow a REST model since most of them are based on simple unidirectional communications that should follow a standard request-response communication model, simplifying its implementation in the process.

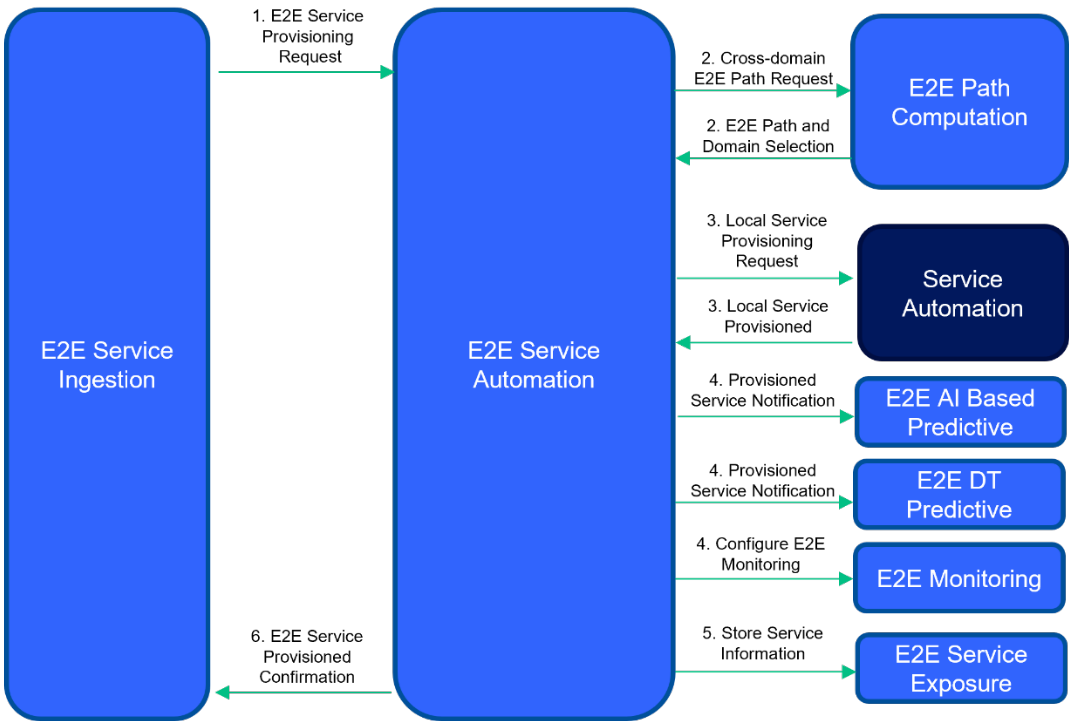

E2E Service Ingestion
1. E2E Service Provisioning Request
6. E2E Service Provisioned Confirmation
E2E Service Automation
2. Cross-domain E2E Path Request
2. E2E Path and Domain Selection
E2E Path Computation
3. Local Service Provisioning Request
3. Local Service Provisioned
Service Automation
4. Provisioned Service Notification
E2E AI Based Predictive
4. Provisioned Service Notification
E2E DT Predictive
4. Configure E2E Monitoring
E2E Monitoring
5. Store Service Information
E2E Service Exposure

Figure 4-11. E2E Service Automation Service provisioning

1. The E2E Service Automation receives a new provisioning E2E service request from the E2E Service Ingestion. The module processes this request to set up a new E2E service in the required management domains.
2. After processing the request, the E2E Service Automation module sends a Cross-domain requests towards the E2E Path Computation service to select a suitable path for the service. Once the paths and the involved domains have been selected by the E2E Path Computation module, the E2E Service Automation module receives both the E2E Path and domain selection information.
3. With the information shared from the E2E Path Computation service, the E2E Service Automation requests a local service provisioning to the MD Service Automation module in each of the selected domains. After a while, the E2E Service Automation module receives a notification to inform that the Local Service has been provisioned in the corresponding MD.
4. After receiving the previous confirmation, the E2E Service Automation sends a New Provisioned Service Notification to both E2E AI-based PD & E2E DT Predictive Analytics services. Furthermore, the E2E Service Automation module sends a Configure E2E Monitoring notification to the E2E Monitoring module.

5. As part of the provisioning service, the E2E Service Automation module instructs the E2E Service Exposure to store the service information in its corresponding module. Since other modules might need this data, and because it could be subjected to modifications depending on the service assurance and its lifecycle, this interaction will be performed using a REST model for its interface.
6. Finally, the E2E Service Automation module sends an E2E Service Provisioned confirmation to the E2E Service Ingestion to confirm the provisioning of the service. This interaction follows a REST model since it is a simple notification to a different module in the AICP. In practice, this communication will be internal, since the E2E Service exposure Ms will be embedded in the E2E Service Automation module.

E2E Service Decommissioning

Figure 4-12 depicts the decommissioning of an E2E service through the AICP from the perspective of the E2E Service Automation. This figure presents the implementation details of all the interactions between the management services involved in decommissioning a previously deployed E2E service.

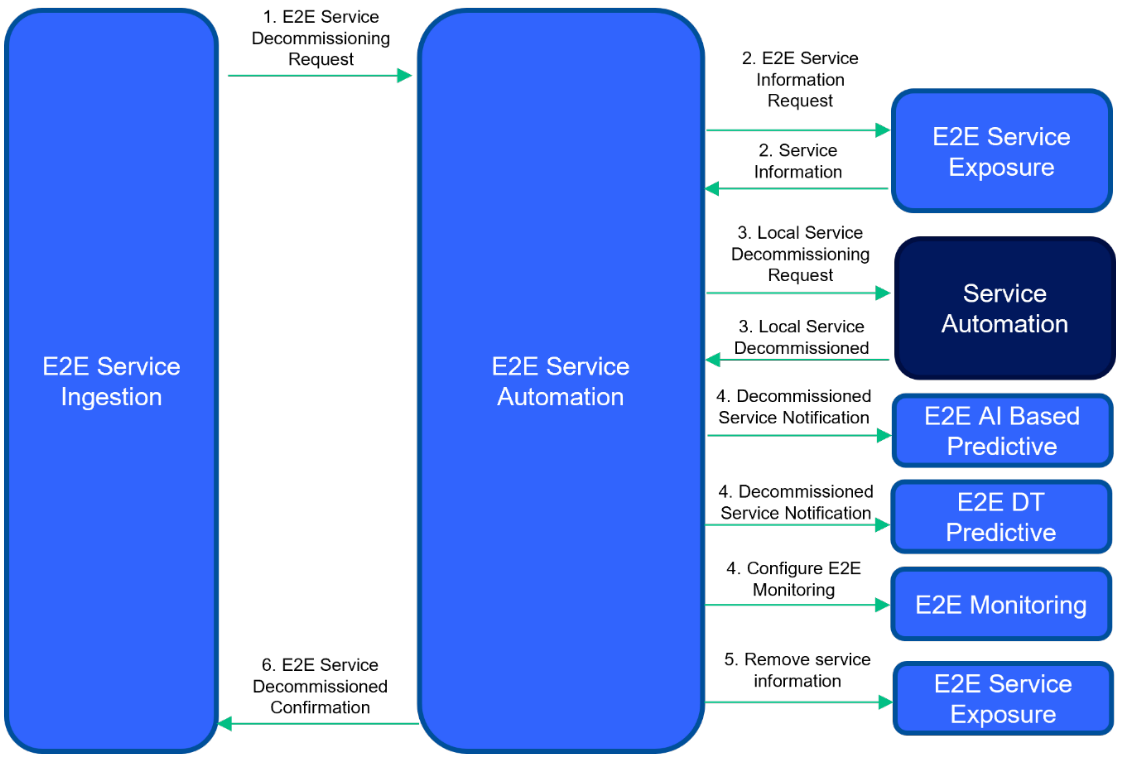

E2E Service Ingestion
1. E2E Service Decommissioning Request
6. E2E Service Decommissioned Confirmation
E2E Service Automation
2. E2E Service Information Request
2. Service Information
E2E Service Exposure
3. Local Service Decommissioning Request
3. Local Service Decommissioned
Service Automation
4. Decommissioned Service Notification
E2E AI Based Predictive
4. Decommissioned Service Notification
E2E DT Predictive
4. Configure E2E Monitoring
E2E Monitoring
5. Remove service information
E2E Service Exposure

Figure 4-12. E2E Service Automation service decommissioning

1. The E2E Service Automation module receives a decommissioning E2E service request from the E2E Service Ingestion. The module processes the request to detect and collect the info needed to remove the E2E service in the corresponding domains.
2. After processing the request, the E2E Service Automation module (internally) retrieves the data of the service to be decommissioned from the E2E Service Exposure.
3. Afterwards, the E2E Service Automation requests a local service decommissioning to the MD Service Automation in each of the involved technology domains. After a while, the E2E Service Automation service receives a notification to inform that the Local Service has been properly decommissioned.
4. After the decommissioning confirmation is received, the E2E Service Automation service sends a Decommissioned Service Notification to both E2E AI-based PD & E2E DT Predictive Analytics services. Furthermore, the Service Automation module also sends a Configure E2E Monitoring notification to the E2E Monitoring module to stop monitoring the decommissioned service.
5. The E2E Service Automation module instructs the E2E Service Exposure to remove the service information. In practice, this communication will be internal, since the E2E Service exposure Ms will be embedded in the E2E Service Automation module.
6. Finally, the E2E Service Automation module sends an E2E Service Decommissioned confirmation to the E2E Service Ingestion to inform about the service decommissioning.

Considering the previous analysis, the proposed SW design for the E2E Service Automation Submodule integrates a set of SW functions, interfaces and a database to cover all the aforementioned functionalities. The specific design and role of each of the proposed components is the following:

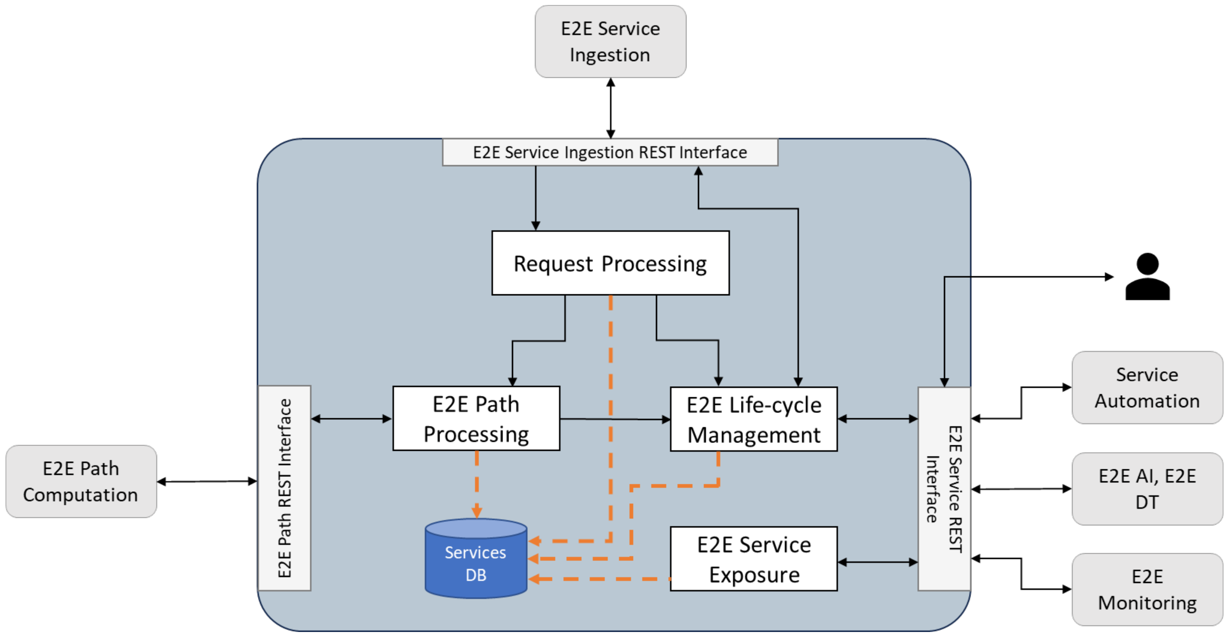


Figure 4-13. E2E Service Automation Submodule – SW Design

E2E Request Processing Function: SW function designed to process and forward the E2E Service Provisioning or Decommissioning requests coming from the E2E Service Ingestion. Among its functionalities, this function is able to process the requests, store or retrieve E2E service information from the Services DB, and forward or receive information from either the E2E Path Processing or to the E2E Lifecycle Management functions depending on the type of workflow. During the Service Decommissioning phase, this function triggers the E2E Lifecycle Management function.

E2E Path Processing Function: SW function designed to manage and process all information related to the path allocation of E2E services. In particular, this function forwards and processes the E2E path computation data via the E2E path interface, forwards the E2E pa.ths and domain information for its storage in the Services DB, and triggers the E2E Service lifecycle management during the Service Provisioning phase.

E2E Lifecycle Management Function: SW function designed to manage the lifecycle of the E2E services. This function is responsible for initializing and enabling the service provisioning and decommissioning towards local domains, notify E2E AI and DT about the service status, request the monitoring of the E2E Service, and forward, request and update information related to E2E Services through its interaction with Services DB. All these interactions are performed through the E2E Service interface.

Funded by the European Union

E2E Service Exposure Function: SW function designed to expose information about the E2E Services. This function is designed to receive exposure requests through the E2E service interface, request the information of E2E Services to the Services DB, and forwarding this information again through the E2E Service interface.

Services DB: Relational database used to store all information related to E2E Services including all characteristics shown in the E2E Service Data Model (see Appendix C, Section 8.3.1) and the link between E2E Services, paths and selected domains. This database is also used by the MD Service Automation SW sub-module to store information related to local services.

E2E Service Ingestion REST Interface: REST interface designed to interact with the E2E Service Ingestion module. This interface is used to receive Service Provisioning or Decommissioning requests from the E2E Service Ingestion, and to forward the outcome, i.e., confirmation or rejection, for those requests.

E2E Path Computation REST Interface: REST interface designed to interact with the E2E Path Computation module. This interface is used to forward E2E path computation requests and to receive path and domain selection information from the aforementioned module.

E2E Service REST Interface: Multipurpose REST interface designed to interact with the Service Automation, E2E AI, E2E DT and E2E Monitoring modules. This interface is used to start the service provisioning and decommissioning in local domains, receives the outcome of those processes, notifies about the status of E2E Services to E2E AI and DT MS, requests the monitoring of a specific E2E Service, and exposes information related to E2E Services to users, operators or other external MS.

The interaction with the interfaces will be performed through the development of specific APIs with different GET, POST, PUT, DELETE methods for each interface. In particular, the set of APIs and methods supported per interface are the following:

Table 4-4. List of APIs exposed by the E2E Service Automation

Interface	Name	Description	HTTP Method	Input/Output

E2E Path Computation	Path Computation Request	Request the computation of the path for a specific E2E service	GET	I: Service description according to the model reported in Appendix C, Section 8.3.1.2 O: A body containing the ID of the E2E service, the ID of the selected path, the selected domains, and the name of the endpoints integrating the path
E2E Service	Local Service Provisioning	Request the provisioning of an E2E service	POST	I: Service description according to the model reported in Appendix C, Section 8.3.1.2, integrating the ID of the local service already initialized. O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	Local Service Decommissioning	Request the decommissioning of an E2E service	DELETE	I: ID of the E2E Service O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	E2E Monitoring Configuration Request	Request the monitoring of the E2E service	POST	I: Service description according to the model reported in Appendix C, Section 8.3.1.2, integrating the KPIs to be monitored. O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	E2E Monitoring Deconfiguration Request	Stop the monitoring of the E2E service	DELETE	I: ID of the E2E Service O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	E2E Service Provisioning or Decommissioning Notification	Notify the outcome of the E2E provisioning or decommissioning request	POST	I: ID of the E2E Service and status O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.

4.4.1.2 MD Service Automation SW Submodule

The MD Service Automation SW Submodule will cover the functionalities and interactions required to perform Local Service Provisioning and Decommissioning.

Local Service Provisioning

Figure 4-14 and **Figure 4-15** depict the workflows where the MD Service Automation is involved for local service provisioning. The workflow for this component is split into two separate flows: (i) the path request loop, where the E2E Path Computation asks the MD to compute an available path for the service, and (ii) the local service provisioning phase, where the remaining interactions of the Service Automation module (at the MD level) are shown:

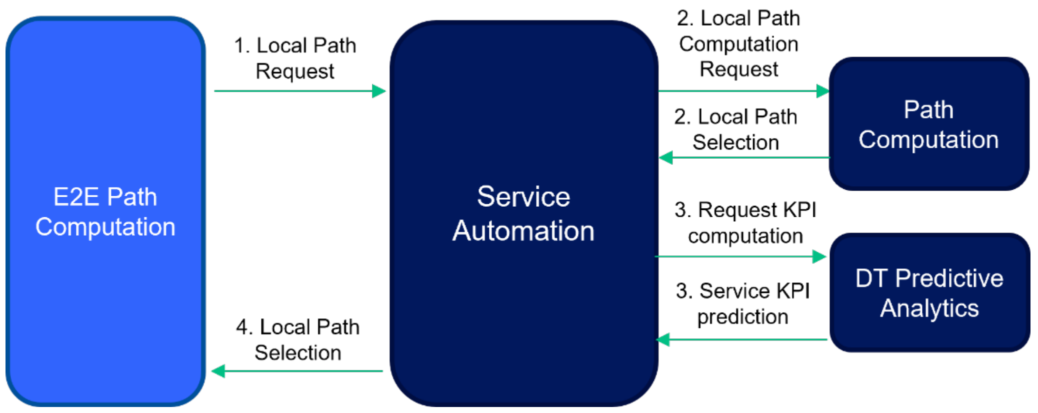


Figure 4-14. Path Request loop in MD Service Automation module

Similarly to the E2E Service Automation module, all service interactions, unless it is specified otherwise, will follow a REST model, given the nature of all the interactions between modules and its lower degree of complexity for its implementation.

1. The MD Service Automation service receives a Local Path request from the E2E Service Path Computation entity. The module processes this request to find an available path in the MD where it is located.
2. After processing the request, the Service Automation module sends a Local Path Computation request towards the MD Path Computation to select a suitable path for the service. Once the paths within each domain have been selected by the MD Path Computation module, that module sends the information about the Local Path to the MD Service Automation.
3. From the information received in the Selected Path, the MD Service Automation requests the computation of the Service KPIs to the MD DT Predictive Analytics. After its processing, the MD Service Automation receives the KPI values from this module.

4. After this process is completed, the MD Service Automation forwards the Local Path Selection to the E2E Path Computation module using a REST interface.

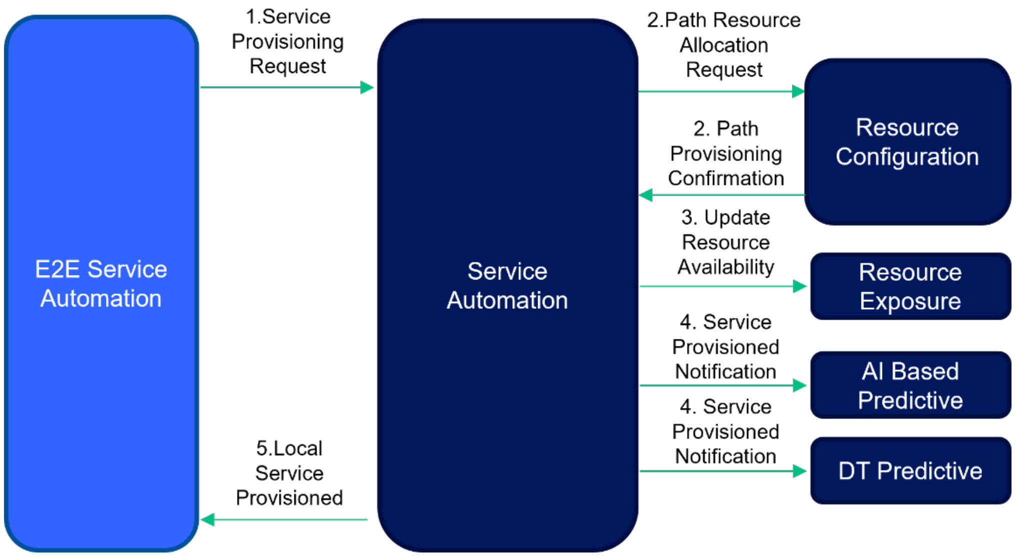

E2E Service Automation
1.Service Provisioning Request
Service Automation
5.Local Service Provisioned
2.Path Resource Allocation Request
2. Path Provisioning Confirmation
3. Update Resource Availability
4. Service Provisioned Notification
4. Service Provisioned Notification
Resource Configuration
Resource Exposure
AI Based Predictive
DT Predictive

Figure 4-15. MD Service Automation service provisioning

1. Once the Path Computation steps have been completed, the MD Service Automation receives the service provisioning request from the E2E Service Automation entity. The module processes this request to set up a new service in its MD with the specifications provided by the E2E component.
2. After processing the request, the MD Service Automation module sends a Path Resource Allocation instruction towards the MD Resource Configuration module to allocate the path resources. Once the resources have been reserved, the Resource Configurator sends a Path Provisioning confirmation module.
3. Once the resources have been allocated in the management domain, the MD Service Automation entity requests an update resource availability to the Resource Exposure Module with the new resource allocation for the selected path.
4. The MD Service Automations sends a service provision notification to both the MD AI Based Predictive and MD DT Predictive services to update the service status.
5. Finally, the MD Service Automation module sends a Local Service Provisioned confirmation to the E2E Service Automation element to confirm the local service provisioning.

Service Decommissioning

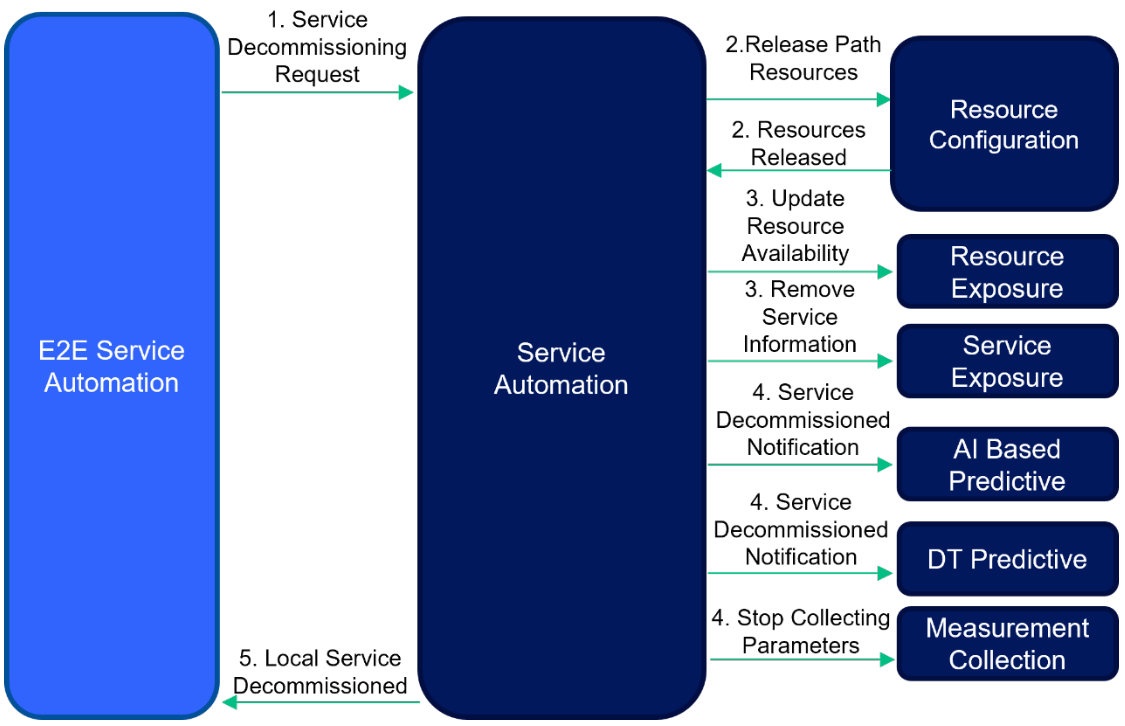

E2E Service Automation
1. Service Decommissioning Request
5. Local Service Decommissioned
Service Automation
2.Release Path Resources
2. Resources Released
3. Update Resource Availability
3. Remove Service Information
4. Service Decommissioned Notification
4. Service Decommissioned Notification
4. Stop Collecting Parameters
Resource Configuration
Resource Exposure
Service Exposure
AI Based Predictive
DT Predictive
Measurement Collection

Figure 4-16. MD Service Automation service decommissioning

1. The MD Service Automation receives a new service decommissioning request from the E2E Service Automation. The module processes this request to remove the service in its management domain.
2. Once the request has been processed, the MD Service Automation module sends a Release Path Resources instruction towards the MD Resource Configuration module to release the path resources. Afterwards, the Resource Configurator sends a Resource Released notification.
3. Following the resource release, the SA module updates Resource Availability in the MD Resource Exposure and removes service information in the MD Service Exposure module, using in both cases their corresponding REST interface. For the MD Service exposure case, this communication will be internal, since the E2E Service exposure Ms will be embedded in the E2E Service Automation module.
4. After the resource update, the MD Service Automations sends a service decommissioned notification to both the MD AI Based Predictive and MD DT Predictive

services to update the service status, as well as notifying the MD Measurement Collection to stop collecting data from the domain.

5. Finally, the MD Service Automation module sends a Local Service Decommissioned confirmation to the E2E Service Automation element to confirm the decommissioning of the local service.

Considering the previous analysis, the proposed SW design for the Service Automation module integrates a set of SW components, interfaces and a database to cover all the aforementioned functionalities. The specific design of this module and the logic connectivity between each component can be seen in **Figure 4-17.** SA module SW design

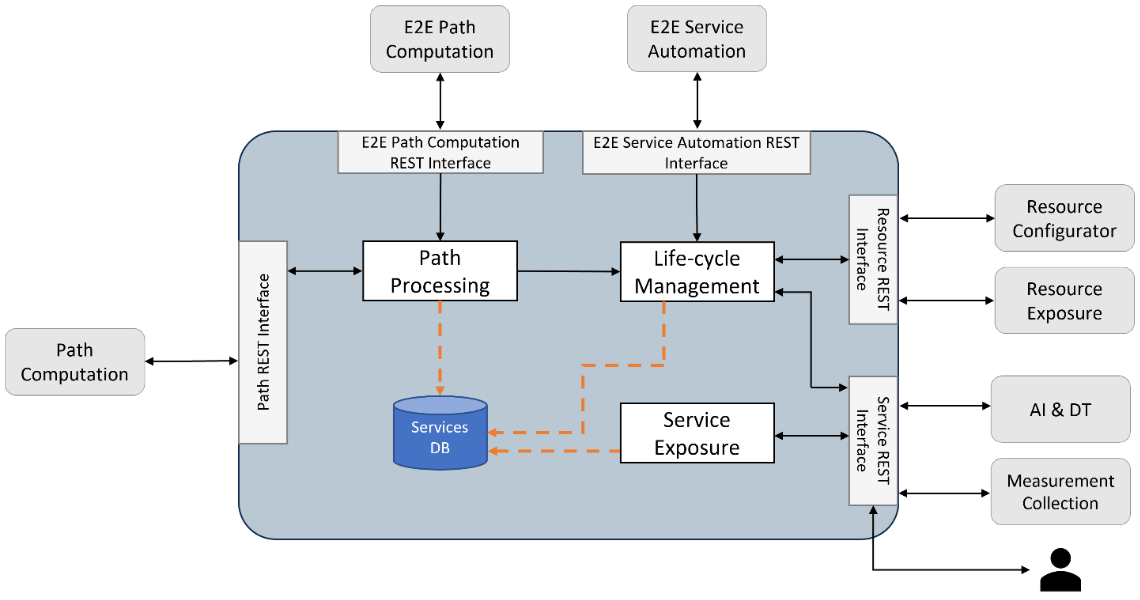


Figure 4-17. SA module SW design

The role of each of the proposed components is the following:

Path Processing Function: SW component designed to manage and process all information related to path allocation in the MD. This component receives the path computation information received from the E2E Path Computation module through its corresponding E2E path interface. Using this data, the component forwards the path request towards the MD Path Computation module to select the path in the domain. Once received, the component proceeds to store the path information in the Services DB, triggering the Service lifecycle management in the process. This component is only used during the service provisioning phase since it can only be triggered when a new path computation request is received.

Lifecycle Management Function: This component is one of the core elements of the module. The lifecycle management component is designed to manage the Service lifecycle at MD level since each E2E domain will be divided into one “local” service (i.e., Services with TD scope) per domain. With this idea in mind, this function is responsible for establishing both service provisioning and decommissioning in the local domains when a new Service commissioning / decommissioning request arrives from the E2E Service Automation through its corresponding interface. This behaviour will be performed by updating its status and resource consumption in the Service DB. Furthermore, the lifecycle management component will contact the Resource configurator to allocate/release the resources used for the Service in the domain, notify AI and DT about service status, and request measurement collection about the Service. For this purpose, two REST interface will be used in the module: one used for resource related requests (Resource REST interface) and one for the remaining interactions (Service REST interface).

Service Exposure Function: SW component designed to expose information about the local Service in the MD scope. This function is designed to receive exposure requests through the Service interface, request the information of the local services to the Services DB, and forwarding this information again through the Service interface.

Services DB: Relational database used to store all information related to E2E Services including all characteristics shown in the E2E Service Data Model (see Appendix C, Section 8.3.1) and the link between E2E Services, paths and selected domains. This database is also used by the MD Service Automation SW sub-module to store information related to local services.

E2E Path Computation REST Interface: REST interface designed to interact with the E2E Path Computation module. This interface is used to retrieve path computation requests from the E2E Path Computation module, as well as forwarding path allocation notifications.

E2E Service Automation REST Interface: REST interface designed to interact with the E2E Service Automation module. This interface is used to receive Service Provisioning or Decommissioning requests from the E2E Service Automation, as well as to forward service commission/decommission notifications.

Service REST Interface: Multipurpose REST interface designed to interact with the Service Automation, AI, DT and Measurement collection modules is utilised to notify about the status of the local Service to AI and DT Mses and start/stop the measurement collection of a Service, as well as to expose information related to the local services to users, operators or other external MSs.

Resource REST Interface: Multipurpose REST interface designed to interact with resource configurator and resource exposure modules. This interface will enable the interaction with the Resource Configurator to allocate the service resources in the domain, as well as to contact the resource exposure when a module or component requires the resources consumed in the domain (present in the Service DB).

Path REST Interface: REST interface designed to interact with the Path Computation module. This interface is used to retrieve path computation requests from the Path Computation module, as well as forwarding path allocation notifications.

The interaction with the interfaces will be performed through the development of specific APIs with different GET, POST, PUT, DELETE methods for each interface. In particular, the set of APIs and methods supported per interface are the following:

Table 4-5. List of APIs exposed by the MD Service Automation

Interface	Name	Description	HTTP Method	Input/Output
E2E Path Computation	Path Computation Request	Request the computation of the path for a specific service	GET	I: Local Service description based on the model reported in Appendix C, Section 8.3.1.3 O: A body containing the ID of the Service and the ID of the selected path.
Path Computation	Path Computation Request	Request the computation of the path for a specific service	GET	I: Local Service description based on the model reported in Appendix C, Section 8.3.1.3 O: A body containing the ID of the Service and the ID of the selected path.
E2E Service Automation	Local Service Provisioning	Request the provisioning of a local service	POST	I: Local Service description based on the model reported in Appendix C, Section 8.3.1.3 O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.

	Local Service Decommissioning	Request the decommissioning of a local service	DELETE	I: ID of the local Service to delete O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
Service	Service Provisioning or Decommissioning Notification	Notify the outcome of the provisioning or decommissioning request	POST	I: ID of the local Service and status O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	Start measurement collection Request	Start collecting measurement for local MD service	POST	I: Local service description, integrating the resources to be monitored. O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	Stop measurement collection Request	Stop collecting measurement for local MD service	DELETE	I: Local service ID O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
Resource	Resource Allocation Request	Allocate the resources of a local Service	POST	I: Local service description, integrating the resources to be allocated. O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	Resource Release Request	Free the resources of a local Service	DELETE	I: Local service ID. O: a proper HTTP code e.g., 200 OK, 404 NOT_FOUND, etc.
	Resource Status Request	Get resource info of local service	GET	I: ID of the local Service, resources to be exposed. O: List of resources used in the local service.

4.4.2 Early Implementation and Release plan

Currently there is no existing implementation of the Service Automation modules, nevertheless, a repository in the project's GitLab is already available. The implementation plan of this module will consist of three releases at M15, M18 and M24, according to the roadmap reported in Section 5. The first release is expected to cover the development of the MD Service Automation SW Submodule, while the second release will include the implementation of the E2E Service Automation Submodule. After those implementation efforts, the second half of the year will be dedicated to adjusting functionalities and performing the integration with the rest of modules. The final release with a functional design and complete integration will be ready in M24.

Regarding frameworks, programming languages and libraries, this module will be developed by using Python as the main programming language. Regarding libraries, FastAPI could be used for the implementation of the different interfaces, where its APIs could be defined by according to the OpenAPI specification. With respect to the Services DB, it could be developed by using PostgreSQL or SQLite solutions.

4.5 Path Computation

The path computation functionality is needed at the two levels defined for the AICP, that is, the end-to-end and the technological domain level. Hence, while the end-to-end path computation calculates an abstract E2E path that traverses a specific sequence of technological domains, which must fulfil the connectivity requirements of the requested service, the path computation component of each technological domain is responsible for computing a low-level path, which is bounded to the underlying data plane technology and whose performance is aligned with the requirements posed by the requested service and the performance offered by the rest of the partial paths that take part of the end-to-end. (e.g., the end-to-end latency requested by the service is split across the different domains involved in the computed path). Therefore, the path computation functionality in the AICP is performed by two types of MSs, namely the E2E Path Computation MS and the Local Path Computation MS. Both MSs are described in (PREDICT-6G/D3.1/7.2.1.4-7.2.2.4, 2023) and participate in the service provisioning process.

The use of the Path Computation MSs is transversal to all UCs, especially during the provisioning phase, as it is the service that provides the specific sequence of network elements (nodes and links) that need to be configured according to the stated KPIs. By engaging with the Exposure services and the DT, it is able to provide path computations at both end-to-end and technological domain (e.g., 3GPP, DetNet, Wi-Fi) level to satisfy the UCs requirements.

Funded by the European Union

4.5.1 SW Design

E2E Path Computation SW module

This sub-section describes the preliminary design and implementation of the software module that will implement the E2E Path Computation MS. Based on the workflows defined in this sub-section first provides some detail on the interactions between the module and the rest of the services, which allows to identify and define the communication interfaces. Afterwards, the functional architecture of the software module is described.

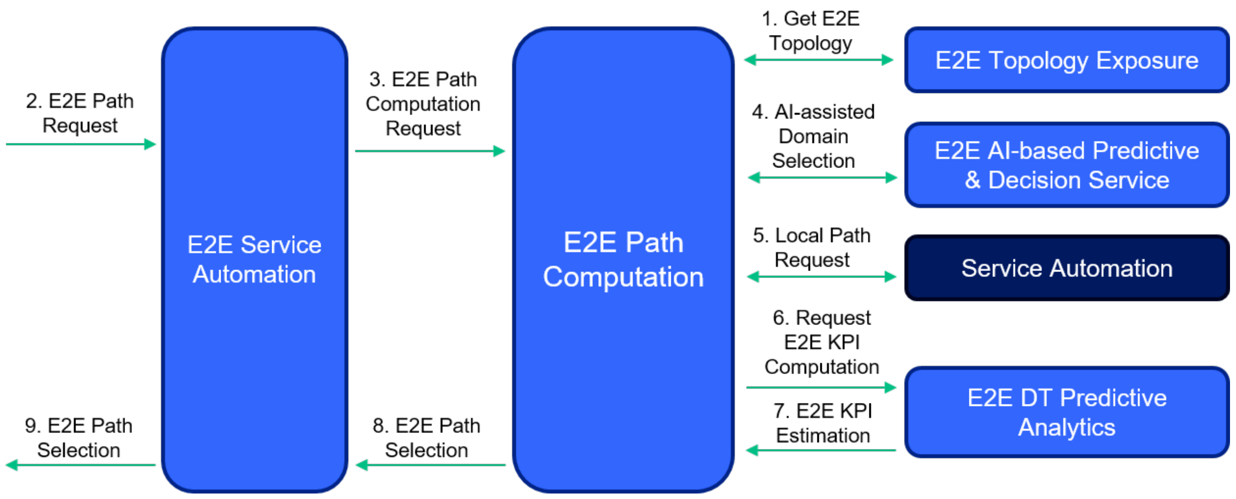


Figure 4-18. E2E path computation request and calculation

Figure 4-18 illustrates the general operation and interaction of the E2E PC module for the service provisioning workflow. First, the PC has to collect the abstracted topological information of the underlying multi-domain data plane (1). To do this, the PC interfaces the E2E Topology Exposure MS. Such information is modelled as a connected graph that contains an abstracted view of the different domains of the data plane as described in Section 4.3. Such abstracted view also contains information about the capabilities and KPIs offered by each domain. During the AICP operation, upon a service provisioning request, the E2E Service Automation MS requests an E2E path to the PC to support such service (2 and 3 in the **Figure 4-18**). The request contains the endpoints that need to be connected as well as the connectivity requirements (e.g., KPIs) that need to be fulfilled. The E2E PC is responsible for computing a valid E2E path over the abstract multi-domain topology and, consequently, for computing the domain sequence to connect the endpoints of the service. To do this, the E2E PC implements an internal logic that considers the information collected from the E2E Topology Exposure MS to compute an E2E abstract path (or list of candidate paths), which fulfils the requirements of the request. Such abstract path, which has been computed considering the KPIs offered by the involved domains, results into a split of the E2E KPIs across them. Hence, each portion of the E2E abstract path is translated to a single-domain path request to the domain-specific Service Automation MS (5). It is worth noting here that the computation engine can be assisted by the AI-based Predictive & Decision MS, which could execute more sophisticated path computation processes (4). Upon the reception of local path request from the E2E PC (5), the Service Automation MS of each involved domain requests a fine-grained local path (or list of paths) computation with the specific KPI requirements for that domain and sends it back to the path computation. Once all the partial paths have been computed, the engine of the E2E PC composes one or more E2E paths that are sequentially sent to the DT (6, 7) for KPI estimation. Afterwards, the E2E PC selects the path that better fulfils the requirements. The selected path is finally sent to the E2E Service Automation MS, which is the responsible for managing the service configuration process.

Figure 4-19 depicts the functional architecture of the E2E PC software module, which is aimed to implement the process described above. Such architecture is composed of five functional blocks:

1. MS Server & Interface: This block implements a REST interface and server devoted to receiving the path requests from E2E Service Automation MS. From here the path computation process is triggered.
2. MS Client Interfaces: This functional block embraces the client interfaces that the module has to implement. In particular, this block contains the interfaces to the Topology Exposure, the AI-based Predictive and Decision MS, the domain specific Service Automation MSs, and the E2E level DT. Said interfaces support the CRUD (Create, Read, Update and Delete) operations of the artefacts related to path computation. To this end, several interface design and technologies can be adopted, such as point-to-point interfaces (e.g., REST, gRPC) or message bus-based.
3. Inventory and Topology: These blocks contain the internal view of the abstracted E2E topology and resources inventory, which will be used by the path computation engine to calculate the E2E path. These elements collect the information from the exposure MS through the corresponding client interface.
4. Path Computation & Decision Engine: This block implements the path computation logic. As said before, the path computation can rely on an internal engine, which considers the topology and inventory, as well as the service requirements conveyed in the path request, to compute a candidate or set of candidate multi-domain E2E abstract paths; but can also rely on AI-assisted path computation mechanisms, which are requested through the client interface to the AI-based Predictive & Decision MS. In addition, this element is responsible for contacting the domain specific Service Automation MSs involved in the E2E abstract path to request the local path

computations (via the corresponding client interface). Afterwards, this block collects the local paths, composes the E2E candidate or set of candidates, and sends it/them to the DT for final KPI assessment. Finally, in case there is a set of candidate E2E paths, this block selects the most appropriated one and sends it back to the E2E Service Automation.

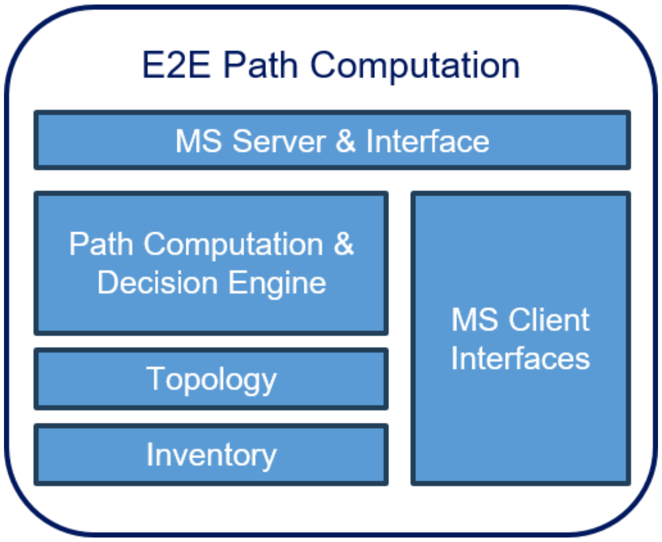


Figure 4-19. Schematic of the internal functional architecture for the E2E Path Computation MS

__Local Path Computation SW module__

As with the case of the E2E Path Computation MS, this sub-section describes the preliminary design and implementation of the technological local Path Computation MS, focusing on the operational workflow that describes the main interaction between the MS and others, as well as on describing the functional architecture of the software module that implements this service.

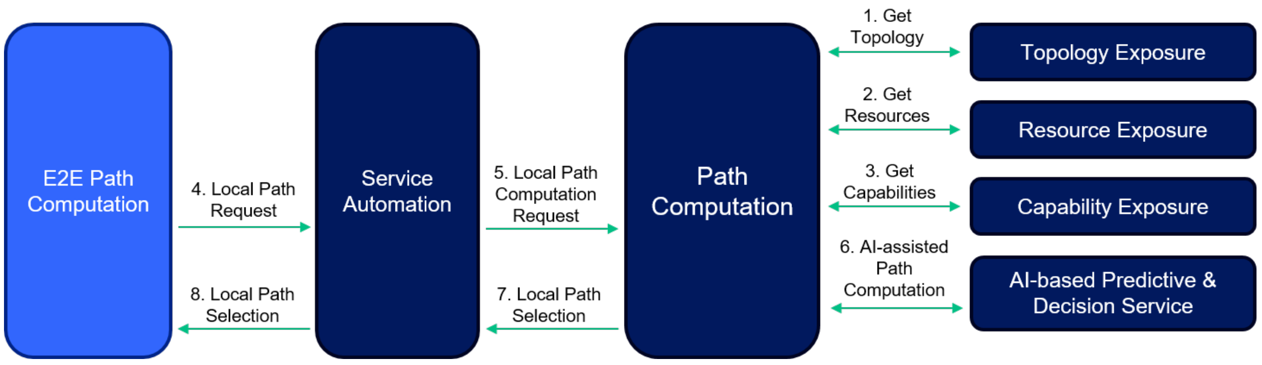


Figure 4-20. Technological domain path computation request and calculation

Figure 4-20 illustrates the operation of the Path Computation MS and its interaction with other MS during the service provisioning workflow. Similar to the E2E case, the module first has to request for the topological details of the technological domain, by interfacing with the Topology Exposure MS at domain level (1). In addition, since fine grain and domain-specific details are required for the low-level path calculation, the service also needs to retrieve the resources of the technological domain and their capabilities to compute the feasible paths according to the concrete technology of the domain. This is done by interfacing with the domain Resource Exposure and Capabilities Exposure MS, respectively (2), (3). With this information, the Path Computation is ready for calculating paths according to requirements posed by service provisioning requests. During said operational workflow, the domain Service Automation MS is contacted by the E2E Path Computation MS to request for the details of the local path, passing as arguments the endpoints of the path as well as the desired KPIs (4). Upon reception, the Service Automation contacts the local Path Computation with a path computation request (5). The local Path Computation executes its own internal logic to determine the candidate paths, which can be assisted by AI functionalities, by interfacing with the AI-based Predictive and Decision MS, to perform a more insightful derivation of candidates (6). Once the candidate paths have been determined, and the list potentially reduced thanks to the AI capabilities. Finally, it sends the list of candidates to the Service Automation (7), which will select the most suitable path to be sent to the E2E Path Computation (8).

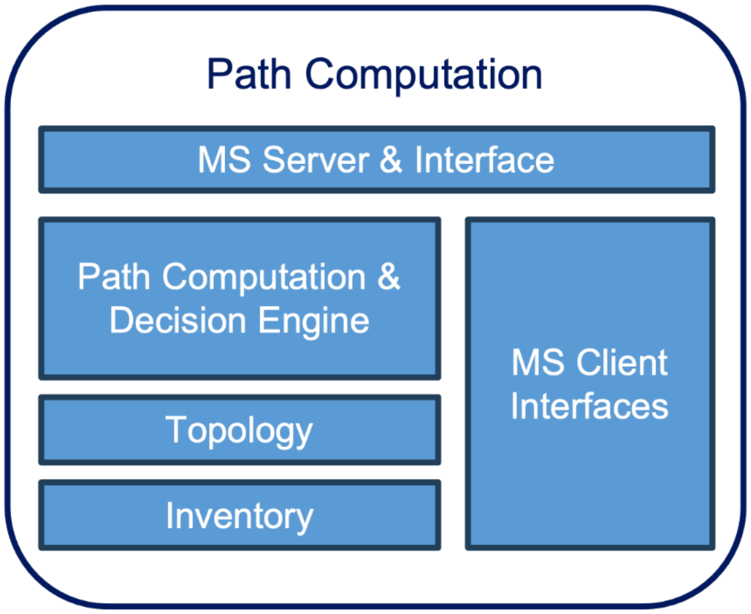


Figure 4-21. Schematic of the internal functional architecture for the E2E Path Computation MS

Figure 4-21 depicts the functional building blocks of the local Path Computation software module, tailored to the implementation of the described workflow. Although having different scopes, the functionalities of the E2E Path Computation and the local Path Computation MSs are fairly similar, hence, their functional building block design is akin. For this reason, in the below list only the differences with respect the E2E Path Computation MS are explained in regard to the functionalities of the local Path Computation MS:

1. MS Server & Interface: This block implements the interface for receiving path computation requests from external modules, like the Service Automation MS.
2. MS Client Interfaces: set of interfaces in order to interact with the MSs needed to construct the topological and resource information of the domain, namely, the Topology Exposure, the Resource Exposure and the Capabilities Exposure MSs.
3. Inventory and Topology: They contain the view of the domain topology, resources and the associated capabilities, employed for calculating the local paths.
4. Path Computation & Decision Engine: This is the block that implements the logic for computing the candidate paths at domain level and decide which is the most suitable path (or paths) that matches the requirements (i.e., the KPIs) stated by the request coming from the Service Automation MS. As said, this decision can be potentially aided by the AI-based Predictive and Decision Service to refine the calculation and/or the selection of paths.

4.5.2 Early Implementation and Release plan

The current version of the Path Computation MS covers the internal path computation logic that determines, given a graph of a network topology, the sequence of nodes and links that conform the path. Said graph is based on a preliminary model topology that covers the details and characteristics of nodes and links, being generic enough to cover both end-to-end and domain level paths. Future releases will evolve said model to specialize the details for the concrete needs of the two types of path computations. Additionally, although not fully developed, it also contains a preliminary set of REST clients to interface with the rest of MSs. The details and operation of the interfaces are still under design. Although for the current deliverable there is no stable software release for the service, a repository in the project's GitLab is currently available. The release plan for the module is intended to be in four phases, following the roadmap reported in this document and the integration and validation tests planed at WP4 level.

The preliminary version of Path Computation MS is being implemented in Java for the internal logic as well as the topology and path inventory. In this regard, Springboot[5] is used as the framework for the implementation of the software module. Together with a set of Maven libraries, said framework is a good candidate that provides the necessary functionalities to develop and implement all the expected requirements. Regarding the interfaces with the rest of the MS to which the Path Computation interacts with, as said previously, the REST paradigm is the main candidate to implement the server interfaces of the module. For the client ones, the module will rely on the communication technologies offered by the other MSs. In any case, Java and Springboot offer libraries that allow to easily define and implement the client and server side of a number of interface technologies (e.g., REST, Websockets, gRPC, etc.), as well as the supported operations and data exchanges.

4.6 Digital Twin

In this section, we briefly describe the main functionalities and requirements of the DT. Then, a preliminary architecture with several modules is proposed. Details about the initial models for TSN traffic to be considered in the DT are presented. Next, those contributions with additional details on the KPI estimation procedures, including the list of measurable KPIs.

[5] https://spring.io/projects/spring-boot

Moreover, the interfaces between DT and TSN CM, as well as KPI computation algorithms are detailed. Finally, the status of the implementation is reported.

4.6.1 SW Design

Figure 4-22 sketches the preliminary architecture envisioned for the DT: it contains the following basic modules: *i)* a manager module configuring and supervising the operation of the rest of the modules; *ii)* a few modules that include algorithms, models, and the interface with path computation MS; and *iii)* a Redis DB that is used in publish-subscribe mode to communicate the different modules among them.

Regarding the CURSA-SQ engine, we adopt an efficient flow-based approach based on a continuous queuing model for network flows analysis. The CURSA-SQ (M. Ruiz at al., 2023) queue model is a continuous G/G/1/k model with a First-In-First-Out (FIFO) discipline based on the logistic function. CURSA-SQ can be applied for a wide range of scenarios, such as generating realistic data for Machine Learning training purposes and for accurate, scalable, and predictive near real-time estimation of end-to-end KPIs in fixed and converged fixed-mobile networks.

The evaluation/tuning and simulation composer modules work together to translate the characteristics and parameters received in the incoming request into a simulation plan that needs to be solved by means of calls to the CURSA-SQ engine. Once the results of such simulations are available in the Redis DB, the KPI estimation module aggregates and post-processes such results to compose the E2E KPIs metrics needed to provide as reply of the received request.

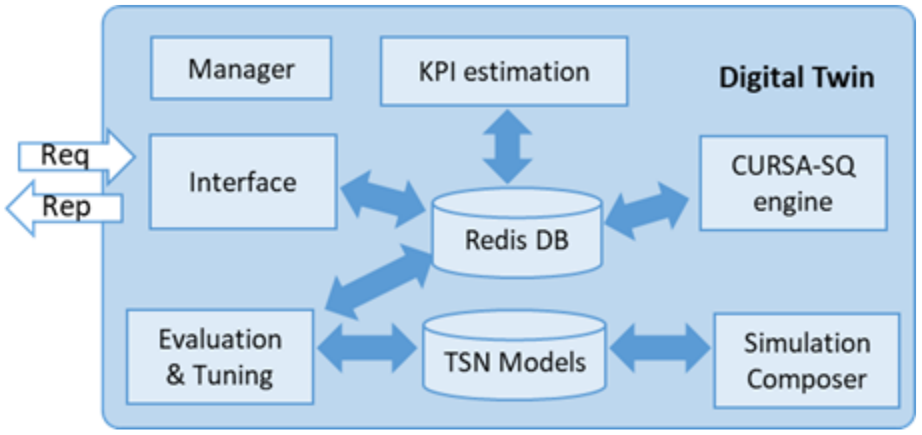


Figure 4-22. DT preliminary architecture

CURSA-SQ extensions to model network interfaces supporting both BE traffic and TSN flows simultaneously under different TSN standards are presented next. Specifically, the following two approaches are initially considered:

1. **Synchronous (*sync*) TSN model**: aligned with IEEE 802.1Qbv standard, time slices are reserved to TSN traffic flows, whereas Best Effort (BE) or QoS committed traffic is transmitted in between consecutive protected TSN slices. Note that this option, although ensures the QoS of TSN flows, might limit that of the BE traffic in the case that the protected time slices are not fully consumed by the TSN traffic.
2. **Asynchronous (*async*) TSN model**: aligned with IEEE 802.1Qcr standard, TSN flows use exactly the transmission time that they need, and inter-packet gaps can be filled with BE frames of suitable size, which maximizes BE throughput and reduces its latency.

To model the mix of both TSN and BE, we assume the scheme of the network interface detailed in **Figure 4-23a**, where *n* individual TSN input flows (XTSN) that arrive conveniently shaped, are combined with an aggregated input BE flow (XBE). Each individual flow *i* is associated to one continuous capacitated queue system, where its state $q_i(t)$ depends on the input flow and on a server rate $\mu_i(t)$ variable with time; all the individual queue systems access the network interface characterized by a fixed server rate μ. To model the sync approach, a time window of fixed length (*T*) is defined and time slices for every TSN flow are reserved; the rest of the time window that remains unassigned can be used to BE traffic (**Figure 4-23b**).

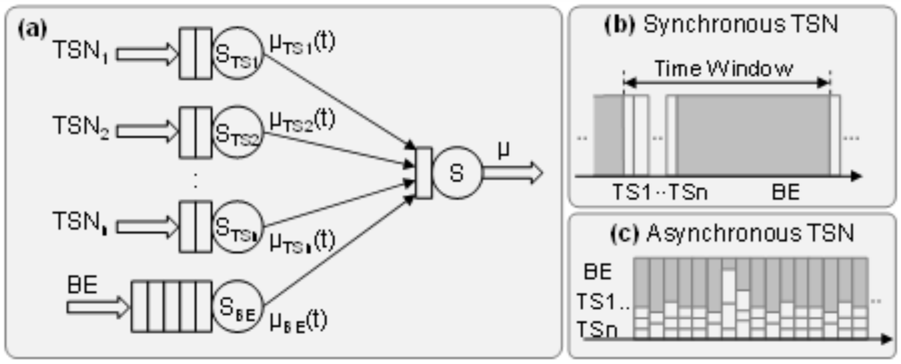


Figure 4-23. TSN Interface model

Figure 4-24 describes with an illustrative example how to model in the DT a simple scenario for QoS KPI evaluation purposes in the event of a new provisioning request. This example involves a factory floor with a TSN traffic flow established between AP1 and the coordinator element (labelled as *f1*), as well as a BE traffic flow that connects users inside the factory with remote contents/services in an external DC placed in the metro network domain (*f2*). Then, a new TSN connection request (*r*) to connect AP2 with the coordinator needs to be evaluated before provisioning. For the sake of simplicity, let us consider that the TSN connectivity manager computes only one candidate route (depicted with a dashed line).

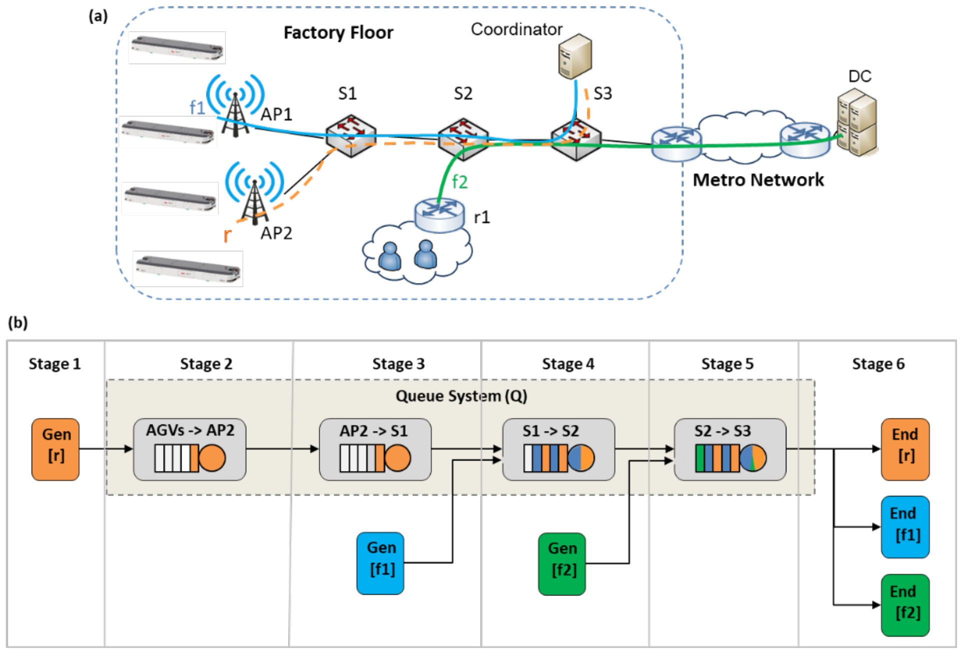


Figure 4-24. Example (a) and its modelling in the DT (b)

Figure 4-24 (b) shows the model of the example in **Figure 4-24(a)** for the UL direction, i.e., from AP/users to coordinator/DC. The model contains three main building blocks:

- **Traffic generator (*Gen*)**: this block generates synthetic traffic to be propagated through the queueing system according to the characteristics of each flow. For both the new connection request *r* and the set of established flows *F*, we assume that traffic characteristics of service *type* are available, by means of either expected models or actual measurements or both. These characteristics include statistical distributions of the following random variables:
    - Inter-packet time
    - Packet size
    - Inter-burst time
    - Burst size

  Synthetic traffic is generated combining the statistical distributions of the abovementioned random variables with other parameters such as *maxTraffic.*

- **Queue system (*Q*)**: Assuming the *sync* queue model approach described above (aligned with IEEE 802.1Qbv standard), time slices are reserved to high priority TSN traffic flows, whereas low priority BE traffic is transmitted in between of consecutive protected TSN slices. Moreover, the sync model emulates other important features such as enhanced preemption (according to 802.1Qbu and 802.3br standards). Characteristics of each queue include speed (in Mb/s or Gb/s) and buffer size (in Bytes), as well as other parameters characterizing time slicing policies.
- **Termination point (*End*):** this module simply serves as sink of the propagated traffic.

The example in **Figure 4-24(b)** shows a model with 4 queues, reproducing each of the involved hops for the request *r*. Specifically, it includes one queue to reproduce the wireless segment (e.g., from an AGVs to an AP), as well as a sequence of queues for each of the wired Ethernet interfaces from AP to the coordinator element. Moreover, the traffic of each of the established flows is injected and terminated before the first interface and after the last one, respectively, of the common segment with *route* of request *r*. Thus, *f1* traffic is injected in the S2 interface that receives traffic from S1 (labelled as *S1->S2*), while *f2* is injected one hop later (in *S2->S3* interface). Both flows terminate after S3.

Assuming the queuing model described above, the following list of KPIs can be measured and quantified by the DT for the flow request under evaluation and the rest of established flows:

- **Throughput** (or data rate): Average volume of traffic (in Mb/s or Gb/s) at the input and/or output of a queuing system at a given time.
- **Traffic loss ratio**: Average percentage of traffic that is rejected at the input of a queuing system due to lack of available queueing capacity at a given time.
- **Delay** (or latency): Average of the elapsed time (in μs or ns) that the traffic experiences to traverse a queuing system from input to output at a given time.
- **Jitter**: Standard deviation of the elapsed time (in μs or ns) that the traffic experiences to traverse a queuing system from input to output at a given time.

For each of the described KPIs, two type of measurements can be obtained:

- **Nominal values** i.e., the actual values that the DT estimates for request *r* once it is considered to be established.
- **Delta values** i.e., the difference (increment or decrement) that the DT estimates comparing the KPIs computed before and after establishing request *r*.

Typically, nominal values can be computed for the request *r*, while delta values are computed for the existing set of flows *F*. However, depending on the specific KPI, service type and use case, the set of available KPI metrics and measurements can change. In any case, the DT is always able to provide relevant and accurate QoS estimation data for the TSN connectivity manager to perform proper decision making.

4.6.2 Early Implementation and Release plan

The version that has been released in this deliverable includes a REST API which implements a preliminary interface. Specifically, this version of the REST API offers several endpoints that provide methods to interact with the Topology DB and to issue the evaluation of the KPI of a selected flow. The REST API offers the following endpoints:

- **NetworkDB:** Allows GET, POST and DELETE methods. Implements the creation, deletion and retrieval of the Topology DB in the Digital Twin.
- **Topology/Node:** Allows GET, POST and DELETE methods. Implements the creation, deletion and retrieval of the Node's information in the Topology DB.
- **Topology/Link:** Allows POST and DELETE methods. Implements the creation and deletion of the Link's information in the Topology DB.
- **Flow:** Allows GET, POST and DELETE methods. Implements the creation, deletion and deletion of the Flow's information in the Topology DB.
- **KPIEvaluation:** Allows GET method. Implements the KPI evaluation of a selected flow of the Topology DB.

This version of the DT includes preliminary algorithms to compute the delay in a set of queues that support a traffic flow. Initial versions of traffic definitions are also included.

4.7 Resource Configuration

Once the requirements of the ingested service are obtained, this module carries out the requisite configuration changes on data plane resources of the pertinent technology domains to meet the service demands. It is composed of per-domain resource configuration sub module.

4.7.1 SW Design

The Resource Management module is the AICP component in charge of configuring the optimal resources of the different domains based on the requirements of the service user wants to allocate.

The module delivers the Resource Configuration Management Service (MS). It receives input from the Service Automation MS and initiates the resource configuration process on identified resources. The Resource Configuration MS realizes the actual updates to data plane resources based on inputs defined in D3.1 (PREDICT-6G/D3.1/7.2, 2023).

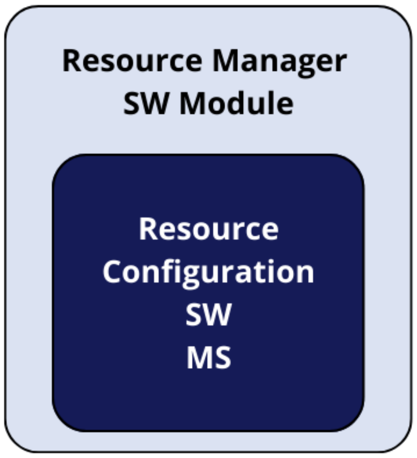


Figure 4-25. Resource Configuration SW Module and Management Service

Resource Configuration management

Figure 4-26 shows the interaction among AICP module for the resource configuration of an ingested service.

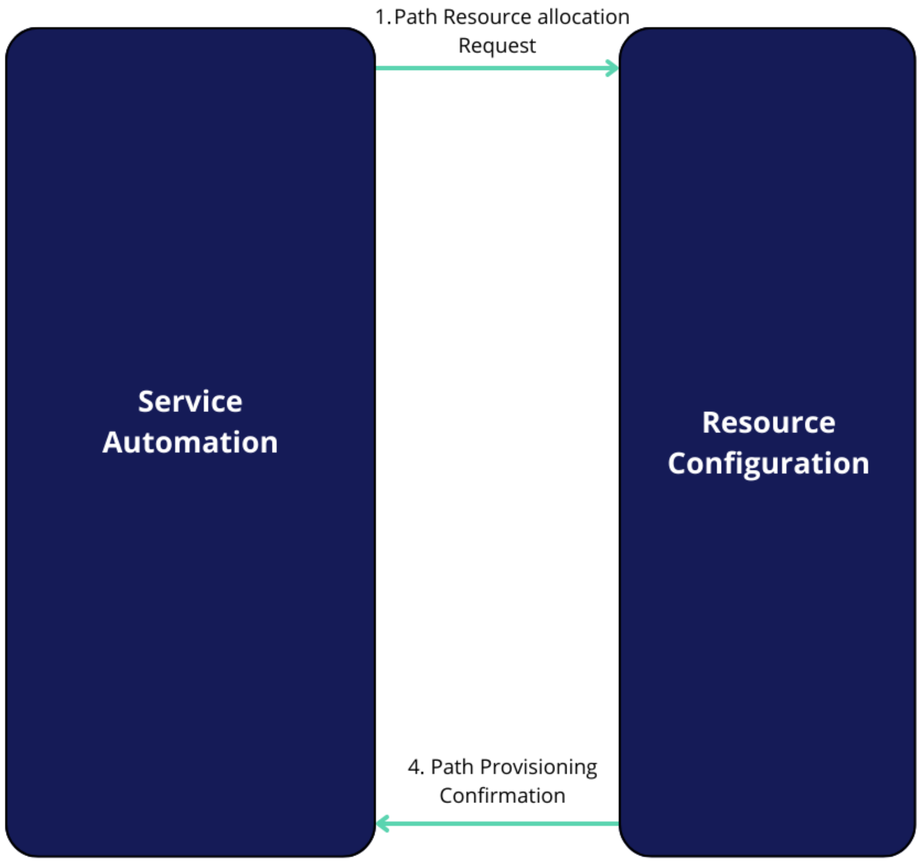

1. Path Resource allocation Request
Service Automation
Resource Configuration
4. Path Provisioning Confirmation

Figure 4-26. Interaction among AICP the modules of the Resource Management for an ingested service

1. Service Automation MS provides as input to the Resource Configuration MS a Path Resource Allocation request. This interaction starts the process of the following exchange of information with the Resource Configuration MS.
2. The process of configure the resources start here. Path information of participating domains is forwarded to the Resource Configuration. Once the configuration is performed, and the resources have been reserved the Resource Configurator sends a Path Provisioning confirmation module, as defined in in Service Automation section.

The Resource Configuration process is illustrated in **Figure 4-27**, detailing the interaction between each MS participant in the process.

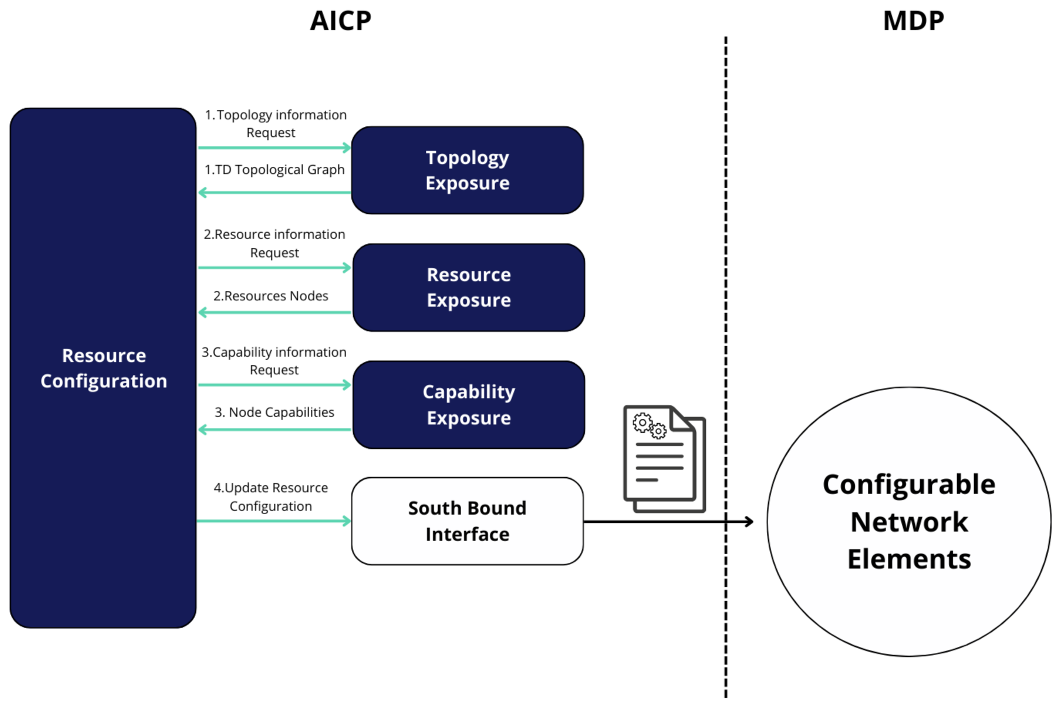


Figure 4-27. Resource Configuration interaction with other MS and the MDP

1. After the arrival of the message starting the resource configuration process, a Topology Information Request is sent to the Topology Exposure MS. This request is replied with a Technological Domain (TD) Topological Graph.
2. When the information from the TD Topological Graph is processed, the Resource Configuration MS will request to the Resource Exposure MS for the information of available resources at each node. After this request, it comes a reply from the Resource Exposure MS with the needed information.
3. In the same way as in “2”, when the previous information is processed, the Resource Configuration MS request to the Capability Exposure MS asking for each node capabilities. The Capability Exposure replies with the needed information.
4. With all the information gathered from the different MS used as inputs, the Resource Configuration MS is ready to generate the needed configuration to achieve the boundaries of the service. When it is done, it will send an update message to the South Bound Interface (to be defined). This module will generate the necessary files to send it to the Data Plane. The insides of the Resource Configuration MS and the South Bound Interface is going to be detailed in followings deliverables.

4.7.2 Early Implementation and Release plan

Work on this software module is ongoing and an initial version is expected in M15 in time for early integration tests as proposed in the roadmap presented in Section 5. The resulting feedback from the integration work will be used to refine the software mainly with regard to the Southbound Interfaces to the Multidomain Data Plane (MDP). A further release is planned with these updates in M18 corresponding to the next integration milestones presented in the roadmap. Although a stable release is currently unavailable, an official repository with work in progress is available in the official GitLab of the project.

4.8 Data collection and management

The Data collection and management module is the AICP component responsible for the gathering of monitoring data from PREDICT-6G data plane, MDP, to feed the logic of analytic processes based on AI/ML and DT for deterministic service monitoring and data-driven network automation decisions.

For this purpose, two specific Management Services (MS) were defined in section 6.2 of D3.1 (PREDICT-6G/D3.1/7.2, 2023) the E2E Monitoring MS and the Management Domain (MD) Measurement Collection MS, with the aim to build a hierarchy of Monitoring Platforms: one local to the technological domains and the one at the E2E level for collecting from domain-specific instances. The goal is to gather metrics from multiple and heterogeneous data sources, through specific plugins whose configuration is dynamically programmable via an open API, to abstract and either to expose in near real-time or to store them in form of timeseries in order to provide them to a data consumer, which can be another AICP SW component, a local application, a human user, etc. The access to the collected data is differentiated on a data consumer basis to prevent a complete open access: so, before reading some kind of data, authentication (through credentials or certificate) and authorization (proper privileges) shall be set. Given the high-level of programmability and the fact that the KPI measurement is crucial for the control of deterministic service, the Data Collection management will be employed to collect data in all technological domain encompassed by MDP and for all UCs of the project.

4.8.1 SW Design

The Data collection and management component is designed to be a modular software to increase the flexibility and maximize the re-usage of existing data collection plugins. Its architecture is represented in **Figure 4-28**.

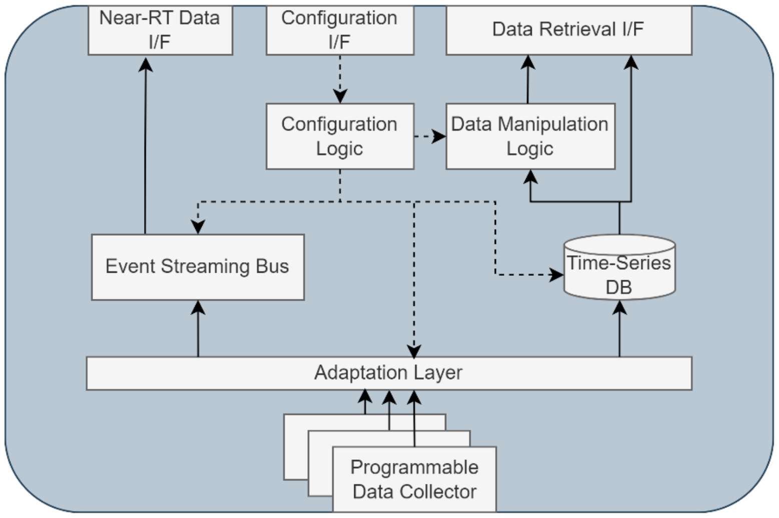


Figure 4-28. Collection and management platform module SW design

The SW architecture present several modules and interfaces:

- **Adaption Layer**. Is the component in charge of exposing in a uniform manner the monitoring data from the heterogeneous data sources. The data sources can belong to the MDP's Technological Domains (TD) or to an AICP MD in form of local monitoring platform, as clarified below. The data are collected through a set of **Programmable Data Collectors** created and terminated on demand (e.g., at service provisioning/decommissioning). The target data source of each collector can be configured at runtime along with sampling frequency, if required. In fact, the collectors can be configured to for either active (poll data sources with a configurable frequency) or passive (data sources push data to the collector) monitoring.
- **Event Streaming Bus**. Is the component in charge to expose data in a near real-time manner i.e., a soon as they are provided by the *Adaptation Layer.* This data is used for continuous deterministic service monitoring, AI/ML inference, and as event to trigger network automation process.
- **Time-series DB**. Is the component in charge to store the data collected (provided by the *Adaptation Layer*) in form of timeseries (historical data collection). This historical data collection can be used for periodic analysis and/or AI/ML models' training.
- **Data Manipulation Logic**. Is the component in charge perform aggregations on the monitoring data (e.g., calculation of min, max, average, standard deviation, etc). This component is optional since the data manipulation functionalities of the related MSs reported in D3.1 (PREDICT-6G/D3.1/6.2.1.2-6.2.2.2, 2023) are not mandatory.

- **Configuration Logic**. Provides configuration functionalities that includes i) creation/termination of *Programmable Data Collectors* and their target data sources at runtime ii) data aggregation and retention time, and iii) data consumer management i.e., access control, authentication and authorization.
- **Multi-technology access interface**. The module exposes 3 different interfaces. The **Configuration I/F** allows the access to the *Configuration Logic* functionalities, exposing REST-based APIs. The **Near-RT** and the **Historic Data Retrieval I/Fs** are designed to access the collected data in near real-time and timeseries respectively. The second one allows the access to aggregated data from the *Data Manipulation Logic,* if available and can be REST or a combination of REST APIs for the aggregated data and the specific timeseries DB interface for the historical data.

4.8.1.1 Hierarchical intra-AICP data collection

As discussed at the beginning of this section, the idea is to implement a hierarchical monitoring structure capable to collect at both MD and E2E MD level. It is worth noting that, from the AICP functional architecture, both the E2E Monitoring MS and the Measurement Collection MS (monitoring service at MD level) share the same functionalities listed below:

- Historical data management
- Real-time data management
- Data Manipulation
- Service Configuration

All of them are covered by the SW design discussed above, so this AICP component can be used both for the local MD and for the E2E MD just by changing the configuration depending on the scope, as shown in **Figure 4-29**.

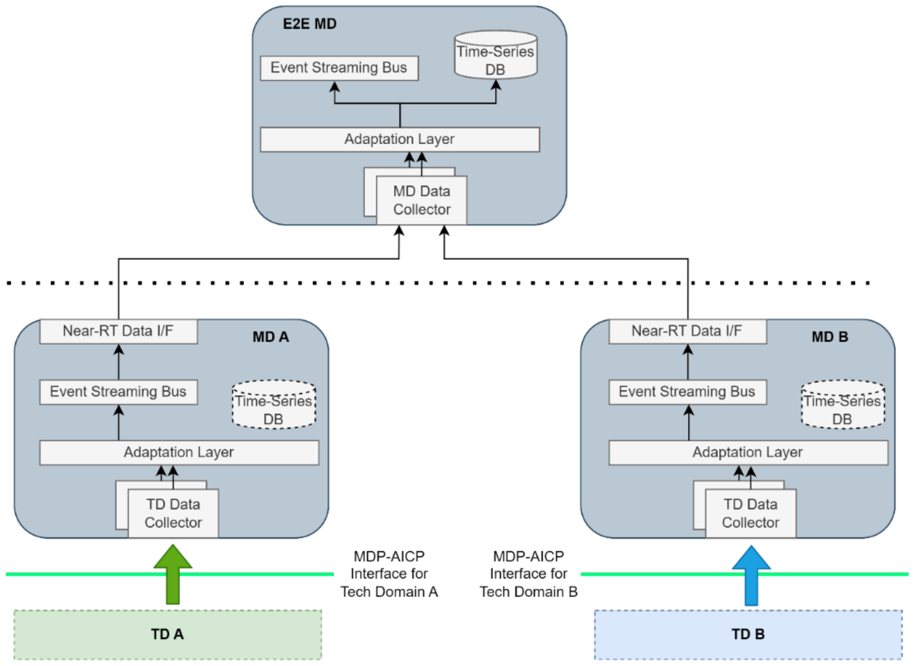


Figure 4-29. Hierarchical configuration of the multiple Data collection and Management components for multi-domain data collection

The figure show that flow of the data from the Technological Domains to the E2E MD. Each local MD monitoring component collects data from its coupled TD through the proper MDP-AICP Interface and uniforms the metrics according to the format:

```
<service_id>, <parameter_name>, <value>, <timestamp>
```

Then, it propagates these translated metrics to the upper layer of the architecture by pushing them to the near real-time bus; the data are also stored in the Time-series DB for local interest, i.e., related to the underlying TD. The E2E MD monitoring component, which aggregates the local MD ones on an E2E service basis, collects the MD data and uniforms them to the format:

```
<e2e_service_id>, <service_id>, <parameter_name>, <value>, <timestamp>
```

Finally, these integral format data are further published both in the bus and in the DB in order to keep track of the service behavior.

4.8.2 Early Implementation and Release plan

The software baseline used to implement the Data Collection and Management in the AICP is the 5G Monitoring Platform built in the context the ANChOR Project[6], a research project funded by the European Space Agency in the context of ESA Artes 4.0 research program. Developed for support AI-driven closed control loops in 5G systems, the 5G Monitoring Platform already covers part of the functionalities described in the software design above and the idea is to evolve it for the purposes of PREDICT-6G. This would include:

- Enlarge the current set of data source supported towards time-sensitive service KPI monitoring.
- Possibility to deploy multiple instances in different domain to enable distributed KPI collection and cross-domain data distribution (i.e., hierarchical monitoring platform).
- Implementation of Authentication/Authorisation mechanisms for differentiated data access control

The platform leverages on some open-source tools, which have been customized, integrated together and extended with additional components developed from scratch. All the internal modules are encapsulated in Docker containers and can be deployed by exploiting container orchestrators such as Docker Swarm[7] and Kubernetes[8].

Figure 4-30 reports the 5G Monitoring platform original SW architecture.

[6] https://connectivity.esa.int/projects/anchor
[7] https://docs.docker.com/engine/swarm/swarm-tutorial/
[8] https://kubernetes.io/

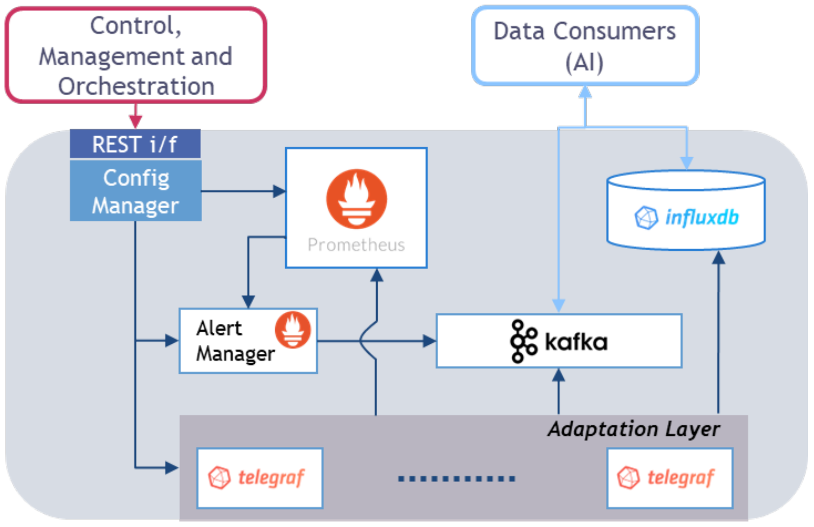


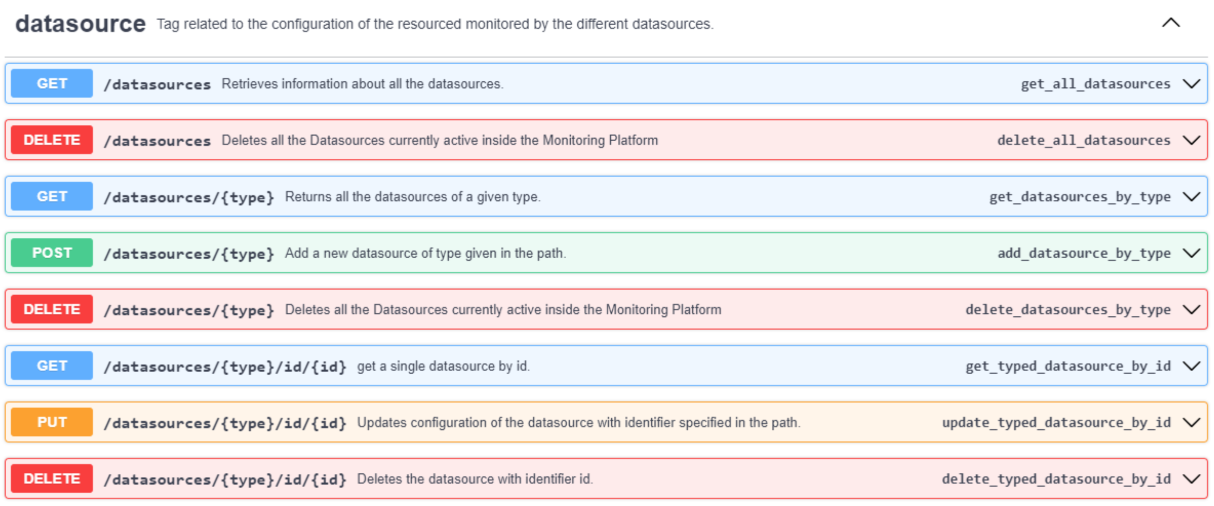
Figure 4-30. Software Architecture of 5G Monitoring Platform

By making a comparison with the SW architecture in **Figure 4-28**, Apache Kafka[9] is used as *Event Streaming Bus* while an instance of InfluxDB[10] covers the role of *time-series DB*. Telegraf[11] is used collect data from the different data sources. In particular, Telegraf allows both passive (listen to data) and active (poll for data) collection from multiple data sources and data post-processing. This implies that a battery of Telegraf instances can cover both the role of *Data Collectors* and *Adaptation Layer*. The *Data Manipulation Logic* and the related interface is implemented with an instance of Prometheus[12], that also offers a tool[13] for setting alarms on data variations (e.g., threshold exceeding). As already discussed, Data Manipulation is an optional functionality (aggregated data e.g., average, standard deviation, etc. are not relevant for time sensitive services) therefore Prometheus will not be employed.

The **Config Manager** is the only component of the platform developed from scratch, covering the role of the *Configuration Logic* and interface. It implements a REST API Server whose YAML specification file has been written according to the OpenAPI specification then exported and extended in Python language. The available endpoints of the interface are divided into two groups: *datasource* for the dynamic configuration of the data sources and *user* for the setup of authentication and authorization of data consumers. The current status of the

[9] https://kafka.apache.org/
[10] https://www.influxdata.com/
[11] https://www.influxdata.com/time-series-platform/telegraf/
[12] https://prometheus.io/
[13] https://prometheus.io/docs/alerting/latest/alertmanager/

configuration for each data source and for each data consumer is persisted in a SQL database implemented through PostgreSQL[14], running on a different Docker container. **Figure 4-31** lists the *datasource* REST API set.



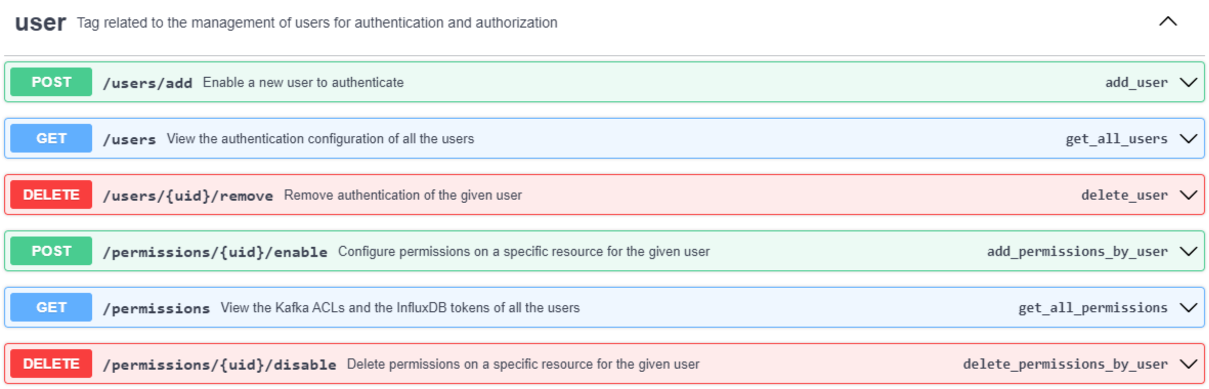
datasource Tag related to the configuration of the resourced monitored by the different datasources.

```
GET     /datasources                   Retrieves information about all the datasources.                                   get_all_datasources
DELETE  /datasources                   Deletes all the Datasources currently active inside the Monitoring Platform         delete_all_datasources
GET     /datasources/{type}            Returns all the datasources of a given type.                                        get_datasources_by_type
POST    /datasources/{type}            Add a new datasource of type given in the path.                                     add_datasource_by_type
DELETE  /datasources/{type}            Deletes all the Datasources currently active inside the Monitoring Platform         delete_datasources_by_type
GET     /datasources/{type}/id/{id}    get a single datasource by id.                                                      get_typed_datasource_by_id
PUT     /datasources/{type}/id/{id}    Updates configuration of the datasource with identifier specified in the path.      update_typed_datasource_by_id
DELETE  /datasources/{type}/id/{id}    Deletes the datasource with identifier id.                                          delete_typed_datasource_by_id
```

Figure 4-31. REST APIs exposed by the Config Manager for the dynamic configuration of data sources

The *user* set of APIs, shown in **Figure 4-32**, is part of the evolution of the platform started in the context of PREDICT-6G and is mainly addressed to system administrators and/or authorized process. When invoked, the Config Manager logic configures authentication and authorization on both Kafka and InfluxDB accordingly.

[14] https://www.postgresql.org/

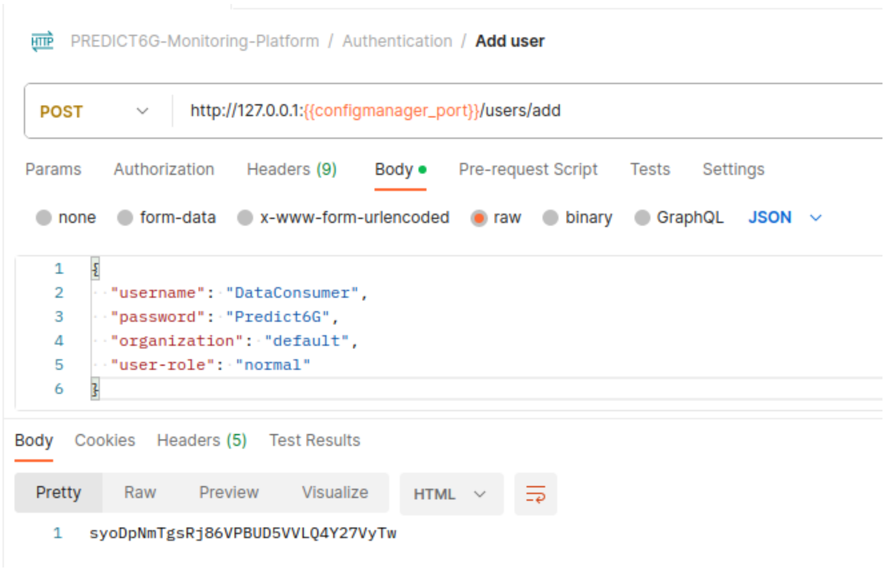


user Tag related to the management of users for authentication and authorization

POST	/users/add	Enable a new user to authenticate	add_user
GET	/users	View the authentication configuration of all the users	get_all_users
DELETE	/users/{uid}/remove	Remove authentication of the given user	delete_user
POST	/permissions/{uid}/enable	Configure permissions on a specific resource for the given user	add_permissions_by_user
GET	/permissions	View the Kafka ACLs and the InfluxDB tokens of all the users	get_all_permissions
DELETE	/permissions/{uid}/disable	Delete permissions on a specific resource for the given user	delete_permissions_by_user

Figure 4-32. REST APIs exposed by the Config Manager for setup of authentication and authorization of data consumers

To correctly access the data related to a service, the consumer requires to be registered to the platform with proper permissions: this happens in two steps, as shown in **Figure 4-33** and **Figure 4-34**, respectively.



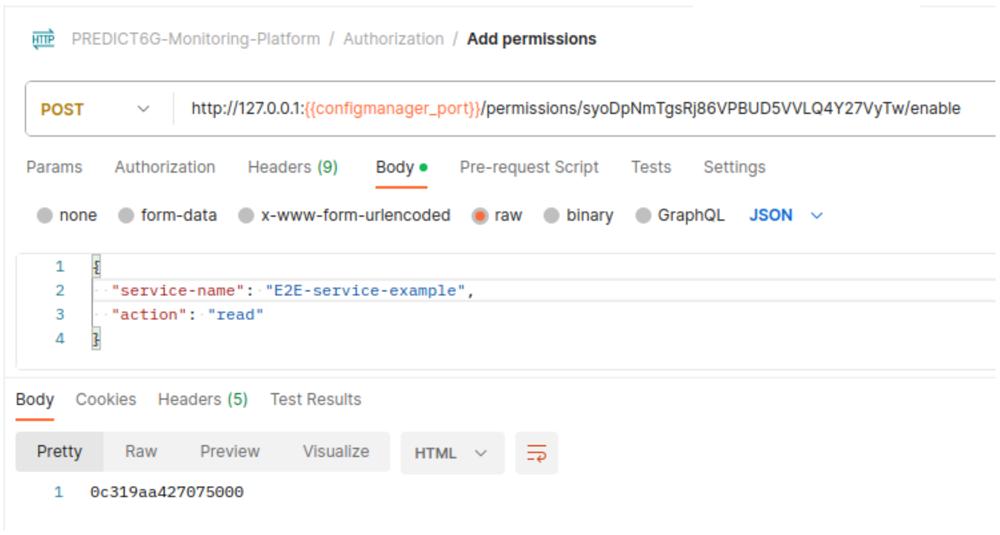
POST http://127.0.0.1:{{configmanager_port}}/users/add

```
{
  "username": "DataConsumer",
  "password": "Predict6G",
  "organization": "default",
  "user-role": "normal"
}
```

```
syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw
```

Figure 4-33. POST request for registering a new data consumer with the credentials provided in the body. The returned value is the Authn-ID uniquely associated to the consumer.

The registration requests return a unique ID (Authn-ID) associated to the consumer. The second request adds permission to the data consumer. Notice the AUTHN-ID in the URL of the request.



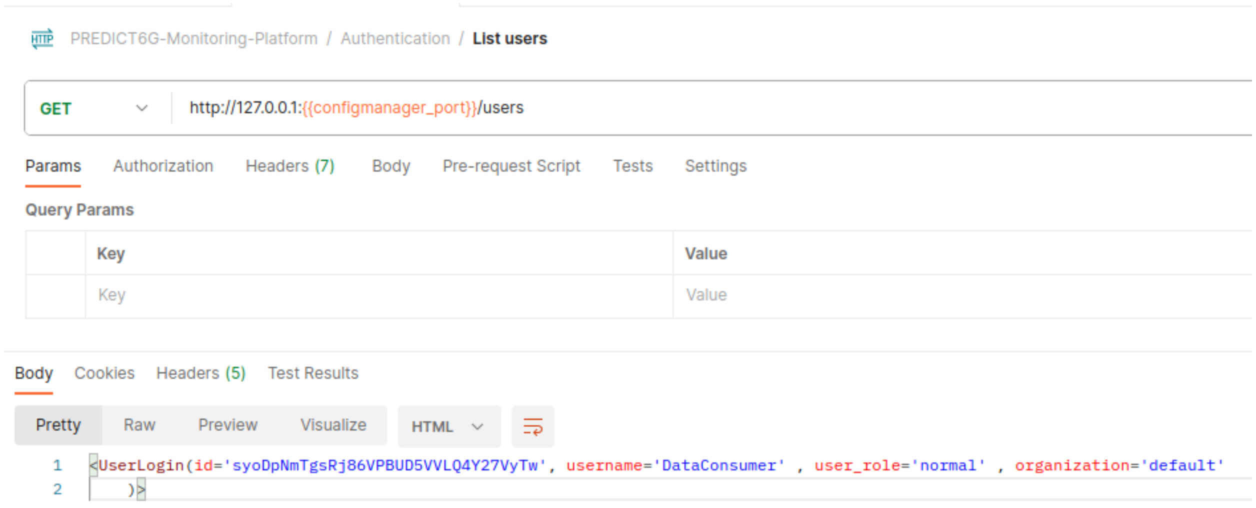
```
POST http://127.0.0.1:{{configmanager_port}}/permissions/syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw/enable
```

```
{
  "service-name": "E2E-service-example",
  "action": "read"
}
```

```
0c319aa427075000
```

Figure 4-34. POST request for authorizing the data consumer with its Authn-ID provided as path parameter and the permissions and the related resource provided in the body. The returned value is the Authz-ID uniquely associated to the consumer

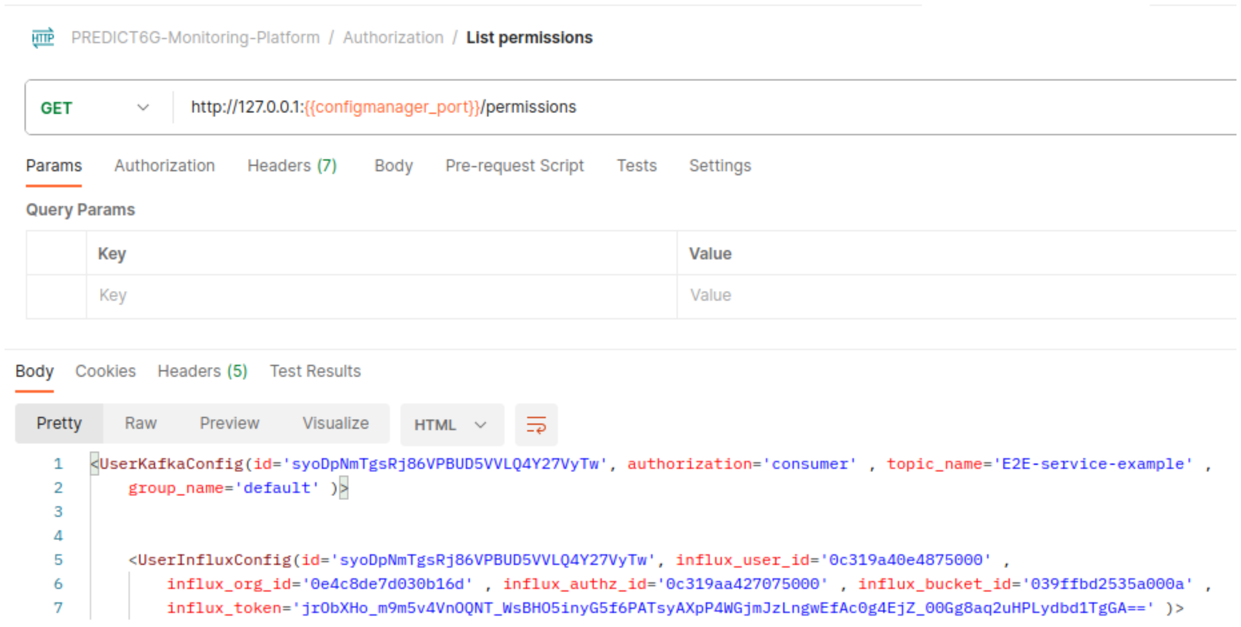
Authentication and authorization information are maintained by both Kakfa and InfluxDB and can be retrieved in any moment through the Config Manager interface as shown in **Figure 4-35** and **Figure 4-36**. The tools can be also directly queried through their own CLI: the related output is quite long, and it is not reported for the sake of readability.


PREDICT6G-Monitoring-Platform / Authentication / **List users**

GET http://127.0.0.1:{{configmanager_port}}/users

```
<UserLogin(id='syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw', username='DataConsumer' , user_role='normal' , organization='default'
    )>
```

Figure 4-35. GET request for listing all the current registered data consumers.


PREDICT6G-Monitoring-Platform / Authorization / **List permissions**

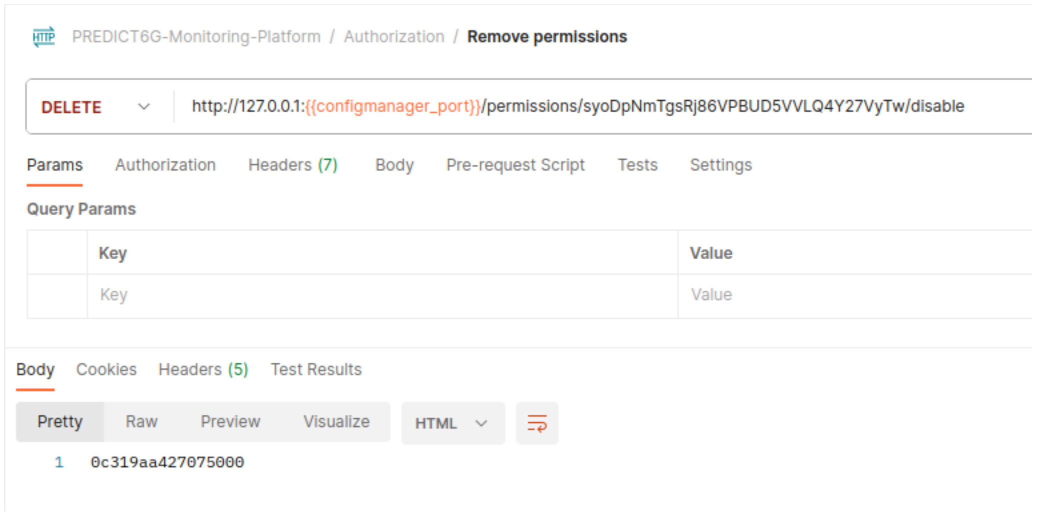
GET http://127.0.0.1:{{configmanager_port}}/permissions

```
<UserKafkaConfig(id='syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw', authorization='consumer' , topic_name='E2E-service-example' ,
    group_name='default' )>


    <UserInfluxConfig(id='syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw', influx_user_id='0c319a40e4875000' ,
        influx_org_id='0e4c8de7d030b16d' , influx_authz_id='0c319aa427075000' , influx_bucket_id='039ffbd2535a000a' ,
        influx_token='jrObXHo_m9m5v4VnOQNT_WsBHO5inyG5f6PATsyAXpP4WGjmJzLngwEfAc0g4EjZ_00Gg8aq2uHPLydbd1TgGA==' )>
```

Figure 4-36. GET request for listing the enabled permissions of all the current registered data consumers

Finally, a data consumer can be removed from the monitoring platform by deleting first its own permissions and then its credentials, through the corresponding request, as shown in **Figure 4-37** and **Figure 4-38**, respectively.



PREDICT6G-Monitoring-Platform / Authorization / **Remove permissions**

DELETE http://127.0.0.1:{{configmanager_port}}/permissions/syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw/disable

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

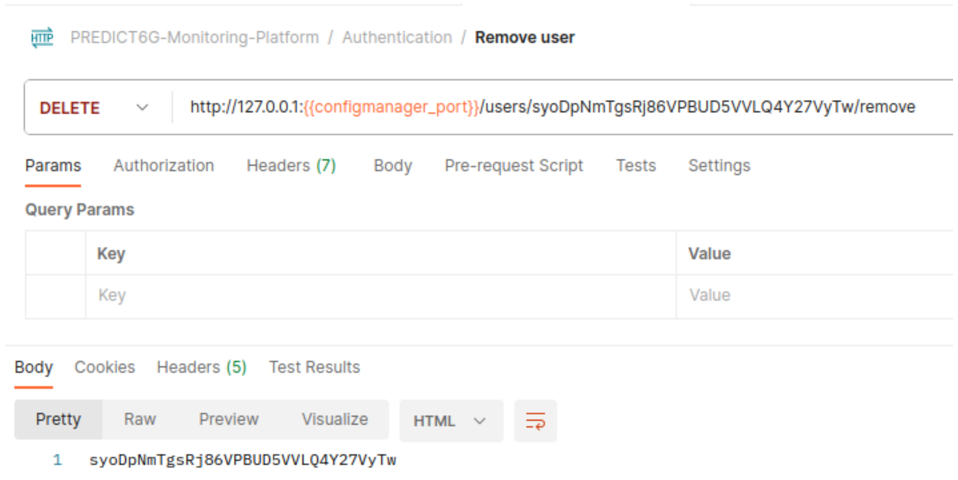
	Key	Value
	Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | HTML

```
0c319aa427075000
```

Figure 4-37. DELETE request for removing the permissions of the data consumer with its Authn-ID provided as path parameter. The returned value is the Authz-ID of the consumer



PREDICT6G-Monitoring-Platform / Authentication / **Remove user**

DELETE http://127.0.0.1:{{configmanager_port}}/users/syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw/remove

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

	Key	Value
	Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | HTML

```
syoDpNmTgsRj86VPBUD5VVLQ4Y27VyTw
```

Figure 4-38. DELETE request for removing the credentials of the data consumer with its Authn-ID provided as path parameter. The returned value is the Authn-ID itself

The source code of already available can be found in the corresponding repository in the project's GitLab:

https://gitlab.netcom.it.uc3m.es/predict-6g/aicp/monitoring-and-data-collection

The next steps for the release plan are:

- Implement the Authentication and Authorization feature also for the Config Manager, in order to prevent that non authorized users can change the configuration. The idea is to leverage on a platform such as Keycloak, which is already mentioned in Section 4.2. This addition will be stable in the release at M21.
- Implement a specific set of data collectors for the different technologies employed by the MDP. Multiple releases are expected in collaboration with WP2 at the end of each development cycle (M15, M18, M21, and M24).
- Cross-instance interactions for hierarchical deployment at M18 with possible refinements in the following cycles.

4.9 AI/ML Algorithmic Frameworks

4.9.1 Orchestration of Machine Learning Tasks Execution

4.9.1.1 Algorithms for optimally distributing learning tasks

Training machine learning (ML) models is notoriously hard, as it requires large quantities of data as well as significant computational resources. To cope with this issue, cooperative training – most notably, federated learning (FL) – has emerged as a universal approach to leverage the resources of multiple nodes to perform a single learning task. In all such scenarios, the data and resources needed to perform the training are scattered throughout different nodes, whose availability and connectivity may significantly vary in both space and time. This results in a major technical challenge, namely, the *mutual adaptation* of the decisions concerning (i) ML training (e.g., model selection and compression), (ii) data selection (i.e., which datasets to use for training), and (iii) node and resource allocation (i.e., at which network nodes to train each portion of an ML model). Several existing works address one or another of the above aspects but fall short of providing a comprehensive strategy to jointly make all the required decisions.

To fill this gap, we developed an algorithmic framework, called Performance-Aware Compression and Training (PACT) (Malandrino et al., 2023), which makes distributed training of deep neural networks (DNNs) sustainable, while being amenable to the presence of heterogeneous nodes with datasets that cannot be shared as well as limited network and computing resources. PACT has the ability to leverage *multiple* DNN models across different stages of the same learning task, by *switching* among them as needed (e.g., through model pruning). It runs at the learning orchestrator and can generate arbitrary stage sequences:

importantly, it optimizes the set of nodes, number of epochs, and model compression along the process. Specifically, for each stage – hence, for each model –, PACT selects the most appropriate datasets, network nodes, and resources. As depicted in **Figure 4-39**, a fairly complex model may be used in the early stages of training, running on a small set of powerful nodes. Later, it is possible to switch to a simpler (e.g., pruned) model, thus including more nodes with less capabilities but more valuable local data. At the same time, the benefits of model switching must be weighed against the cost of switching itself, which requires additional resources and will often result in a (temporary) drop in learning performance.

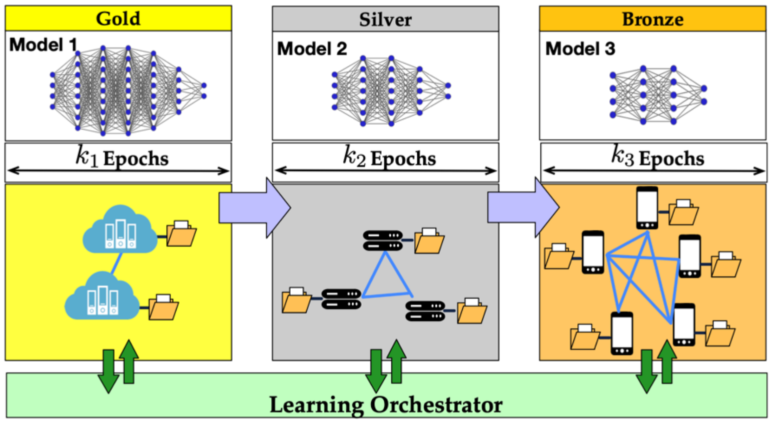


Figure 4-39. Distributed training process of an AI/ML model. Subsets of nodes sequentially train compressed versions of an original DNN model. Nodes are categorized based on their capability and data availability (gold, silver, bronze).

4.9.1.1.1 The PACT algorithmic framework

The goal of PACT is to let the learning orchestrator efficiently find high-quality solutions to the problem of distributed learning illustrated in **Figure 4-39**, which, as we proved in (Malandrino et al., 2023), is an NP-hard problem. PACT consists of three main steps:

1. Create an *expanded graph* representing the possible decisions that the Learning Orchestrator can make, and their outcome;
2. Using such a graph, identify a set of decisions deemed *feasible,* based on the estimated value of loss that would be obtained by applying such decisions;
3. By combining learning- and energy-related information, choose the best feasible solution to enact.

Step 1: Expanded graph. The expanded graph is a directed graph built according to the following rules: (i) The vertices represent the states of the system; they are labelled with the current epoch k, model m(k) and set of nodes n(k) being used, and the total elapsed time T(k) and current loss value l(k). With the aim of identifying feasible solutions, the latter quantity is computed using the robust estimators λchange(k, m, m′) and λrun(k, m, n) that can be experimentally obtained as detailed in (Malandrino et al., 2023); (ii) Elapsed time and loss values are represented, respectively, with resolutions γT and γl (e.g., if γl = 0.1, a vertex with l = 0.1 or 0.2 can exist, but not with l = 0.15); (iii) A directed edge is drawn between two vertices if there is an action (i.e., selection of model and node cluster to execute it) making the system move from one corresponding state to the other; each edge is labelled with the energy consumption of the associated action; (iii) Each vertex representing a feasible state of the system (i.e., with l(k) ≤ lmax and T (k) ≤ T max) is further connected to a virtual node Ω through a zero-cost edge.

The graph is created through the CREATEEXPANDEDGRAPH function, presented in Alg. 1 in **Figure 4-40** (the notation used therein is summarized in **Table 4-6**).

First, all *vertices* are created, representing all valid combinations of model and set of nodes, epoch, loss value, and elapsed time (Lines 3–6). Note that the quantization parameters γ_l and γ_T (Lines 4–6) allow controlling the trade-off between size of the graph and quantization error. For each vertex v, the effect of taking action a from vertex v is determined by computing the resulting elapsed time and the required energy (Lines 12–13). If either is infinite, then taking action a while in state v is impossible, and we move on to the next action. Otherwise, the loss l′ resulting from taking the action is computed using the robust estimator (Line 16). Now, tuple $(k+1, m', n', l', T)$ would describe the state the system lands on after performing a from v; however, due to the way the vertices are created (i.e., using γ_l and γ_T), such a tuple may not correspond to any vertex. Accordingly, in Lines 17–18, l′ and T' are cast into integer multiples of γ_l and γ_T. Then, vertex v' representing the new state is identified (Line 19), and an edge from v to v' is added using the appropriate energy value E as its weight. Finally, if v is feasible, v is connected to Ω (Line 22).

Algorithm 1 Creating the expanded graph

```
function CREATEEXPANDEDGRAPH
    $\mathcal{V} \leftarrow \{\Omega\}$                                  ▷ set of vertices
    for all $m \in \mathcal{M}, n \in \mathcal{N}$ do
        for all $k \in \left[1, 2, \dots, \left\lceil \frac{T^{\max}}{\gamma_T} \right\rceil\right]$ do
            for all $\ell \in [0, \gamma_\ell, 2\gamma_\ell, \dots, \ell(0)]$ do
                for all $T \in [0, \gamma_T, 2\gamma_T, \dots, T^{\max}]$ do
                    $v \leftarrow (k, m, n, \ell, T)$
                    $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
    $\mathcal{E} \leftarrow \emptyset$                                    ▷ set of edges
    for all $v = (k, m, n, \ell, T) \in \mathcal{V}$ do
        for all $\mathbf{a} = (m', n') \in \mathbf{A}$ do
            $T' \leftarrow T + \tau^{\text{change}}(m, n, m', n') + \tau^{\text{run}}(m, n)$
            $E \leftarrow \epsilon^{\text{change}}(m, n, m', n') + \epsilon^{\text{run}}(m', n')$
            if $T' > T^{\max} \vee E = \infty$ then
                continue                        ▷ infeasible, skip this action
            $\ell' \leftarrow \ell + \hat{\lambda}^{\text{change}}_{\text{rob}}(k, m, m') + \hat{\lambda}^{\text{run}}_{\text{rob}}(k, m, n')$
            $\ell' \leftarrow \gamma_\ell \left\lceil \frac{\ell'}{\gamma_\ell} \right\rceil$
            $T' \leftarrow \gamma_T \left\lceil \frac{T'}{\gamma_T} \right\rceil$
            $v' \leftarrow (k + 1, m', n', \ell', T')$
            $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v, v', \text{weight} = E)\}$
        if $\ell \leq \ell^{\max} \wedge T \leq T^{\max}$ then
            $\mathcal{E} \leftarrow \mathcal{E} \cup \{v, \Omega\}$                ▷ feasible state
    return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
```

Algorithm 2 Finding feasible paths

```
function FINDFEASIBLEPATHS
    $v_{\text{curr}} \leftarrow (k, m, n, \ell, T)$
    $\mathcal{P} \leftarrow \emptyset$                                   ▷ feasible paths
    for all $v$: $(v, \Omega) \in \mathcal{E}$ do
        $p \leftarrow \textbf{shortestPath}(v_{\text{curr}}, v)$
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{p, \text{weight} = \sum_{e \in p} \text{weight}[e]\}$
    return $\mathcal{P}$
```

Algorithm 3 Choosing the next action

```
function CHOOSEACTION
    scores $\leftarrow \{\}$
    for all $p \in \mathcal{P}$ do
        $\overline{w} \leftarrow 0$                                      ▷ opportunity
        Le $\leftarrow 0$   ;   Lr $\leftarrow 0$
        for all $((k, m, n, \ell, T), (k', m', n', \ell', T')) \in p$ do
            Le$\leftarrow$Le$+\hat{\lambda}^{\text{change}}_{\text{exp}}(k, m, m')+\hat{\lambda}^{\text{run}}_{\text{exp}}(k, m, n')$
            Lr$\leftarrow$Lr$+\hat{\lambda}^{\text{change}}_{\text{rob}}(k, m, m')+\hat{\lambda}^{\text{run}}_{\text{rob}}(k, m, n')$
        opp $\leftarrow$ Le/Lr
        $\overline{\mathcal{V}} \leftarrow \{v \in \mathcal{V} \colon v[1] = m\}$                        ▷ risk
        Wr$\leftarrow \min_{\overline{v} \in \overline{\mathcal{V}}}$ weight(shortestPath($p[1]$,$\Omega$,via $\overline{v}$))
        risk $\leftarrow$ Wr/weight[$p$]
        scores[$p$] $\leftarrow$ weight[$p$] $\cdot$ risk/opp
    $p^\star \leftarrow \arg\min_{p \in \mathcal{P}}$ score[$p$]
    return $\mathbf{a} = (p[1][1], p[1][2])$
```

Figure 4-40. PACT algorithmic framework.

Symbol	Description
k	Current epoch
K_j	number of epochs to execute before j-th model switching
F_j	pruning factor at j-th model switching
$\mathcal{M}$	Set of all possible models
$\mathcal{N}$	Set of all available nodes
M	Cardinality of $\mathcal{M}$
N	Cardinality of $\mathcal{N}$
$m(k)$	Model used at epoch k
$n(k)$	Sets of nodes used at epoch k
$\ell(k)$	Test loss value at epoch k
$\Delta\ell(k)$	Variation in loss function yielded by epoch k
$E(k)$	Cumulative energy consumption until epoch k
$\Delta E(k)$	Energy consumed to perform epoch k
$T(k)$	Time at which epoch k finishes
$\Delta T(k)$	Time taken by epoch k
τ^{change}	Time to switch the model or nodes
τ^{run}	Time to execute a given model over a set of nodes
ϵ^{change}	Energy to switch the model or nodes
ϵ^{run}	Energy to execute a given model over a set of nodes
λ^{change}	Loss variation when switching model
λ^{run}	Loss variation during one epoch of training
$\ell^{\max}$	Target loss value
$T^{\max}$	Time limit
$\ell^{ideal}(k)$	Ideal loss time at time t
$\mathbf{s}(k)$	State at epoch k
$\mathbf{a}(k)$	Action at epoch k
$\mathbf{A}^k$	Feasible set of actions at epoch k
$\mathbf{C}(\mathbf{s}(k), \mathbf{a}(k))$	Cost function expressing the cost of executing action $\mathbf{a}$ while in state $\mathbf{s}$ at epoch k
$\mathbf{V}(\mathbf{s}(k))$	Value function, i.e., how desirable it is to be in state $\mathbf{s}(k)$
$\hat{\lambda}^{change}_{exp}(\hat{\lambda}^{run}_{exp})$	Expected-value estimator of $\lambda^{change}(\lambda^{run})$
$\hat{\lambda}^{change}_{rob}(\hat{\lambda}^{run}_{rob})$	Robust estimator of $\lambda^{change}(\lambda^{run})$
γ_l	Loss resolution of expanded graph
γ_T	Time resolution of expanded graph
$\mathcal{G}=(\mathcal{V},\mathcal{E})$	Expanded graph
$\mathcal{V}$	Set of vertices of expanded graph
$\mathcal{E}$	Set of edges of expanded graph
v	Vertex in expanded graph
Ω	Virtual node to which each vertex representing a feasible state of the system is connected through a zero-cost edge
$\mathcal{P}$	^{2} Set of feasible paths
p	Path $\in \mathcal{P}$
`opp`	Opportunity factor
`risk`	Risk factor
`score`	Score

Table 4-6. Notations used in Algorithms 1, 2, and 3 of the PACT framework.

Step 2: Feasible paths. Next, PACT uses the expanded graph to identify a set of paths deemed *feasible*; the first edge of such paths represents a feasible action. To mitigate the impact of potential errors in the loss estimation (which in principle may jeopardize feasibility), the expanded graph is built using the *robust* estimators of the loss variation, which guarantees that all paths landing at a feasible node are, indeed, feasible with high probability. Thus, using function FINDFEASIBLEPATHS in Alg. 2 in **Figure 4-40** (the notation used therein is summarized in **Table 4-6**), PACT seeks for paths that (i) start from the current state, and (ii) arrive to a feasible state, i.e., to a vertex connected to Ω. Specifically, for each vertex v corresponding to a feasible state, it determines the shortest path (Line 5) from the current state v_{curr} to v. Such paths are collected in set P and associated with a weight corresponding to the sum of weights (i.e., energy consumption) of their edges.

Step 3: Making the best decision. Once the set of feasible paths, and associated feasible actions, has been identified, using robust estimators to choose the decision to enact would be overly cautious, possibly resulting in unnecessarily higher energy costs. Thus, PACT accounts for two additional aspects when selecting an action: an *opportunity* and a *risk* factor. Such factors and the path weight are integrated into a *score*, and the action corresponding to the lowest score is enacted. For every path $p \in P$, scores are computed in the CHOOSEACTION function in Alg. 3 in **Figure 4-40** (the notation used therein is summarized in **Table 4-6**).

The opportunity factor, $opp \geq 1$, is given by the ratio of (i) the sum of the expected loss to (ii) the sum of the robust loss associated with the edges in p (Line 9). The intuition is to make it easier to choose actions with a good expected loss, since the robust estimator may be too pessimistic. As for the risk factor, its high-level purpose is to avoid undoing decisions. To this end, PACT seeks for paths on the expanded graph that lead from the first node of p, to a vertex $v \in V$ associated with the current model m (Line 10), and thence to Ω. The risk factor, $risk \geq 1$, associated with path p is then computed in Line 12 as the ratio of the minimum among the weights of such paths to the weight of p (defined in Alg. 2). The score of path p is obtained in Line 13 as p's weight, divided by the opportunity factor, and multiplied by the risk factor. Then the action associated with the minimum-score path is returned. It is important to underline that the shortest path going from the current state to Ω represents the lowest-cost decision since edge weights are set to the energy cost of the corresponding actions. Thus, the ultimate outcome of this step is the action with the lowest energy cost to enact.

4.9.1.1.2 Performance results

We assess PACT's performance focusing on a smart factory-based application in a sequential learning scenario, like the one depicted in **Figure 4-41**, where models are passed among individual nodes. We consider the VGG 19 model for image classification, using the CIFAR-10 dataset, and three nodes, each belonging to a different category, namely, gold, silver or bronze. They have respectively 17,500, 12,500, and 7,500 samples from the CIFAR-10 dataset. While the bronze node has a balanced data distribution, the gold and the silver ones have unbalanced datasets: the gold node has 2,750 for each of classes 1–6 and 250 for each of classes 7–10; the silver node has 1,500 samples for each of classes 1–8 and 250 for each of classes 9–10; finally, the bronze node has 750 samples per class. The training time and energy values used for the gold, silver, and bronze nodes reflect (resp.) the capabilities of NVIDIA Ampere A100, NVIDIA RTX A4000, and Raspberry Pi's Videocore 6 GPUs. Finally, we set the target loss to l^{max} = 1.05 and, for simplicity, a very long-time limit of T^{max}=1,000 time units.

We compare the performance of PACT against the following benchmarks: (i) Optimum: the optimal decisions yielding the minimum cost, found through brute-force search and using the true loss evolution; (ii) NoSwitch: no model switching occurs, meaning that only the full model is used; (iii) OneSwitch: only two models are used. For both the NoSwitch and the OneSwitch solution, we consider the best decisions they yield for each value of l^{max}. Specifically, for OneSwitch, we consider the lowest energy cost, feasible strategy changing once, considering all combinations of models and changing epochs. Note that most state-of-the-art works (Gou, J. et al., 2021)(Gao, Z., et al., 2021)(Zhang, T., et al., 2021) envision pruning once, hence, their performance would be represented by OneSwitch.

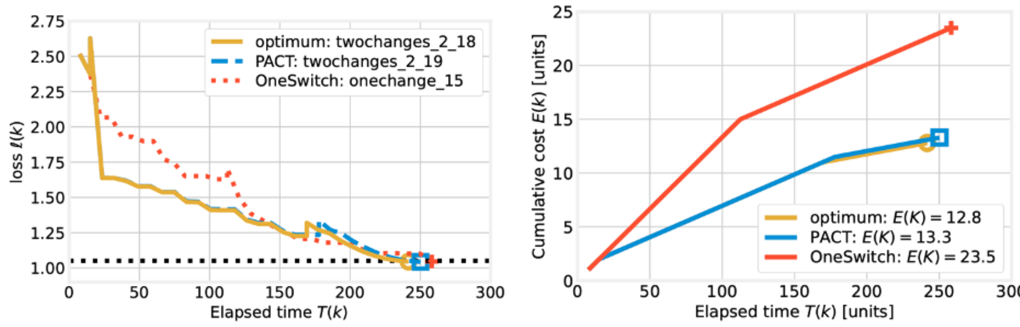


Figure 4-41. Loss (left) and total energy cost (right) as functions of the training time

We evaluate PACT's effectiveness, i.e., how the cost (consumed energy $E(K)$) it yields compares to that of the benchmarks. To this end, Figure 17(left) depicts the time evolution of loss $l(k)$ for l^{max}= 1.05. Note that the peak due to the loss variation λ^{change} incurred when switching models is not always present, as the testing loss can decrease even when switching the model. This is especially true when the first switching is performed early, relative to the PACT and the optimal solutions. Remarkably, PACT makes virtually the same decisions as the optimal policy, i.e., performs the model switching at (almost) the same times. OneSwitch can only switch once, hence, does so later. Interestingly, all the strategies achieve the learning target at almost the same time. However, we recall that the total energy cost is the optimization objective, while time is a mere constraint. Accordingly, **Figure 4-41**(right) highlights how the optimum indeed takes slightly shorter than PACT to reach the objective and does so at a (marginally) lower cost (see the position of the last marker on the y-axis). On the other hand, OneSwitch solution, despite taking a comparable time, requires much more energy to reach I_{max}.

4.9.1.2 Cooperative learning across multiple administrative domains

In the current world, where data is distributed along multiple entities, training machine learning models encounters well known problems. On one hand, the exposure of the data from one entity to the others may entail privacy issues if the data is susceptible. On the other hand, transporting all necessary data may suppose a great amount of network and computational resources, which translate into high energy consumption.

In addition, data generated inside a unique domain may not be enough to properly train the model. Hence, data from other administrative domains must be exploited to increase the quality of the trained model.

Centralized schemes take advantage of a single optimized computational node to execute the model training over it. However, transporting all data to that point can be an arduous task, as a high amount of network resources is needed.

Decentralized Learning (DL) solves this problem. It is based on training the same model in multiple entities, partially collecting the data (e.g., using only its data), and then gathering and joining trained models from the surrounds of each entity. Repeating actively this process ensures that data from all the system influences in the model.

However, DL face the challenge of identifying the correct models to collect from the neighbourhood to be integrated together, as wrong decision may cause catastrophic results and useless models. In (Yang et al., 2021), they propose a DL architecture called E-TREE Learning where nodes are grouped in clusters to aggregate their model weights together, and then follows a hierarchical approach that allows different frequencies among the branches to end up with a global model. To make these clusters, they also present a grouping algorithm that considers the datasets of the nodes.

In the aforementioned scenario, where a set of administrative domains cooperate towards the accomplishment of a training task, our goal is to define a DL logical training topology, which specify the interactions among the administrative domains. At the same time, we want to select the resources employed by each administrative domain to fulfil the task, from the computing nodes and data sources to the amount of computational and network resources, and even exploiting the resources from other administrative domains through resource federation. These decisions are taken with the objective of minimize the energy consumption of the overall process.

4.9.1.1.3 Architecture of the system

We consider a system with multiple administrative domains composing a Federated Domain Set (FDS), coexisting with the purpose of taking advantage of each other's resources. To this end, they are connected through a Federation Layer, managed in each domain by the Federating Orchestrator (FO) – see **Figure 4-42**. Each administrative domain owns two datasets for a given task, one for training and one for testing. The latter is continuously updated with input data collected within the administrative domain; thus, at any time instant it reflects the current distribution of input data and possible drift that may emerge.

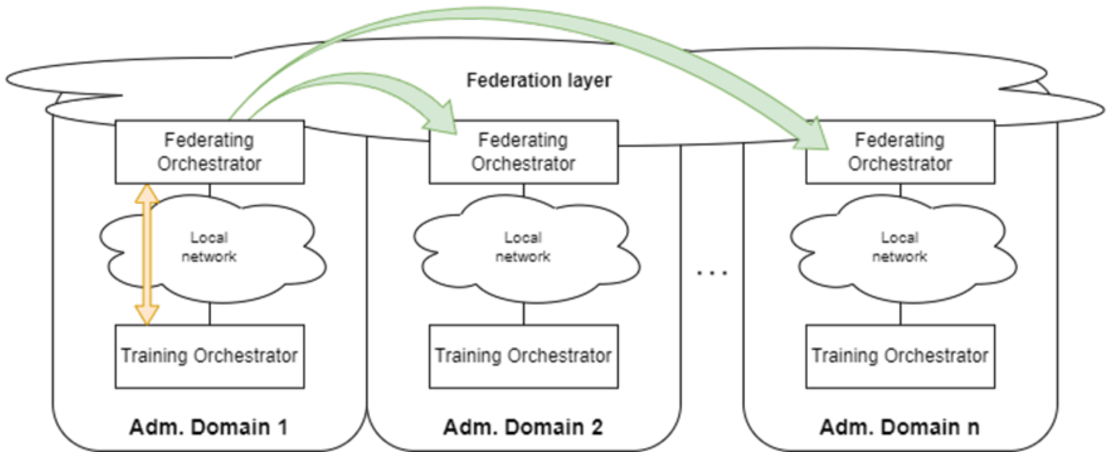


Figure 4-42. Scheme of the architecture of the system.

Within each administrative domain, there is also a Training Orchestrator (TO), which, whenever a model needs to be trained, asks the FO of that domain for the following information:

1. List of active trustworthy administrative domains (obtained through the Federation Layer);
2. Topology of the network connecting the administrative domains;
3. Topology of each administrative domain (only involved nodes and their connections);
4. Type of data: data distribution and related metadata (e.g., if the traffic belongs to dense or sparse population locations);
5. Type of resources (e.g., computing, network, energy) and availability within each administrative domain;
6. Communication delay (or available bandwidth) between administrative domains.

The last four bullet points are retrieved (directly from the administrative domains composing the FDS) using a smart contract developed by the FO.

With this information, the TO decides the nodes participating in the training, how they should interact, and the resources needed to execute the model training.

4.9.1.1.4 Interactions between administrative domains. Decentralized Learning.

Remember that the administrative domains will cooperate to train a model following a DL scheme, so we are going to review how it works and how the model weights are updated.

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected simple graph representing the physical topology of the network connecting the administrative domains $\mathcal{V}$. As mentioned, each administrative domain $v \in \mathcal{V}$ owns two datasets that should not be shared with others, one for training $D_v^{(train)}$ and the other for testing $D_v^{(test)}$. While locally training a model within an administrative domain, the model weights $\mathbf{w}_v$ are updated as follows:

$$\mathbf{w}_v = \mathbf{w}_v^{old} - \eta \nabla_{\mathbf{w}} L\left(\mathbf{w}_v^{old};\ D_v^{(train)}\right)$$

where η is the learning rate and $L\left(\mathbf{w}_v^{old};\ D_v^{(train)}\right)$ is the training loss. Executing the training requires $l_{model}|D_v^{(train)}|$ operations, being l_{model} the executed operations for training the model with one sample, so it is model-dependant.

Administrative domains exchange the locally-computed weights with their neighbours on the logical topology. Let $G' = (\mathcal{V}, \mathcal{E}') \subseteq G$ be a spanning subgraph of G representing the logical topology, and $A_{G'} = \{a'_{i,j}\}_{i,j \in V}$ its adjacency matrix, for

$$a'_{i,j} = \begin{cases} 1 & \text{if } i \in \mathcal{N}^{G'}(j) \Leftrightarrow j \in \mathcal{N}^{G'}(i) \\ 0 & otherwise \end{cases}$$

Hence, an administrative domain $v \in \mathcal{V}$ updates its own weights using those from its neighbours, i.e.,

$$\mathbf{w}_v = \sum_{i \in \{v\} \cup \mathcal{N}^{G'}(v)} \lambda_{v,i} \mathbf{w}_i$$

where $\mathcal{N}^{G'}(v)$ is the set of neighbours of v on the graph G', and $\lambda_{v,i}$ is the aggregation weight for neighbour i of v; in our case, $\lambda_{v,i} = 1/(1 + \sum_{j \in V} a'_{v,j})$, but other solutions can be considered. The number of operations to perform the aggregations depends on the number of neighbours sending weights $|\mathcal{N}^{G'}(v)|$ and the size of the weights $l_{\mathbf{w}}$. Also, $\mathbf{te}_{v,n}$ is the energy wasted on transmitting a byte from $v \in \mathcal{V}$ to $n \in \mathcal{N}^{G}(v)$.

4.9.1.1.5 Measuring data distributions

The accuracy of the model strongly depends on the input data that is used to train and the output that it produces, so one can relate differences in the datasets with differences in the accuracy. One idea to measure these differences in datasets is by comparing their distributions, either input or output. These continuous distributions have one dimension per feature of the samples.

In classification problems, each sample of the input training data is marked with a ground truth label belonging to a class. Then, we can reduce the distribution accounting only for the class labels, being now a discrete distribution of one dimension, and use the Euclidean distance to measure it (Snell et al., 2017).

Let $\bar{\mathbf{d}}_v$ and $\bar{\mathbf{d}}_w$ be the normalized class distributions for the administrative domains $v, w \in \mathcal{V}$, one can measure the differences between them with the Euclidean distance $\left\|\cdot\right\|_2$,

$$\left\|\bar{\mathbf{d}}_v - \bar{\mathbf{d}}_w\right\|_2 \coloneqq \left(\sum_{i=1}^{|\bar{\mathbf{d}}_v|} \left(d_{v,i} - d_{w,i}\right)^2\right)^{\frac{1}{2}}.$$

4.9.1.1.6 Modelling the system

Our solution will minimize the energy consumption during model training. To this end, we need to characterize this process and model the components.

Each administrative domain $v \in \mathcal{V}$ has a set of nodes $\mathcal{V}^v$ with R computational resources to fulfill the local training. Let $\mathbf{R}_v = \{r_{i,j}\}$ and $\mathbf{R}_v^{max} = \{r_{i,j}^{max}\}$ be the $|\mathcal{V}^v| \times R$ matrices representing the used and maximum amount of resource j of node i, respectively. Additionally, node $i \in \mathcal{V}^v$ owns a portion of the total training dataset $D_{v,i}^{(train)} \subseteq D_v^{(train)}$. Notice that $r_{i,j}^{max} = 0$ if node i cannot provide resource j, and $D_{v,i}^{(train)} = \emptyset$ if node i does not own any data sample. Let also $\mathbf{tn}_v = \{tn_n\}_{n\in\mathcal{V}^v}$, where $tn_n \in \{0,1\}$ is a Boolean decision representing whether node n performs training or not.

One can construct a graph $G^v = (\mathcal{V}^v, \mathcal{E}^v)$ that captures the logical data movement among the nodes within the administrative domain $v \in \mathcal{V}$; where $\mathbf{A}_{G^v}$ is its adjacency matrix. This data movement is represented by the bandwidth and the number of transmitted bytes. Let $\mathbf{B}_v$, $\mathbf{B}_v^{max}$ and $\mathbf{TB}_v = \{tb_{i,j}\}$ for,

$$tb_{i,j} = \begin{cases} \left|D_{v,i}^{(train)}\right| & if\ tn_i = 0, \left|D_{v,i}^{(train)}\right| > 0 \\ l_w & otherwise \end{cases}, \quad i,j \in \mathcal{V}^v.$$

be the $|\mathcal{V}^v| \times |\mathcal{V}^v|$ matrices representing the assigned percentage and maximum bandwidth, and the number of bytes transmitted for the edge between each pair of nodes, respectively.

Local model training within the administrative domains can be divided in three steps:

i. Data extraction: Extracting data samples from the storage is an energy consuming process that depends on the amount of data samples and the efficiency of the node $\mathbf{ee}_v = \{ee_n\}_{n\in\mathcal{V}^v}$. Then, the consumed energy at extracting the data $\zeta\left(\mathbf{ee}_v, D_v^{(train)}\right)$ within an administrative domain $v \in \mathcal{V}$ is

$$\zeta\left(\mathbf{ee}_v, D_v^{(train)}\right) \coloneqq \left\|\left\{\left|D_{v,i}^{(train)}\right|\right\}_{i\in\mathcal{V}^v} \circ \mathbf{ee}_v\right\|_{1,1},$$

where $\|\cdot\|_{1,1}$ denotes the entry-wise matrix norm, and $\mathbf{A}$ ⍰ $\mathbf{B}$ the Hadamard (or element-wise) product between matrices $\mathbf{A}$ and $\mathbf{B}$.

ii. Data transmission: Let $\mathbf{ME}_v \in \mathbb{R}^{|\mathcal{V}^v|\times|\mathcal{V}^v|}$ be the matrix representing the energy of moving a byte between two nodes. Hence, the consumed energy while transporting the data $\xi(\mathbf{A}_{G^v}, \mathbf{ME}_v, \mathbf{TB}_v)$ within an administrative domain $v \in \mathcal{V}$ is

$$\xi(\mathbf{A}_{G^v}, \mathbf{ME}_v, \mathbf{TB}_v) \coloneqq \|\mathbf{A}_{G^v} \circ \mathbf{ME}_v \circ \mathbf{TB}_v\|_{1,1}.$$

iii. Computing actions: As reviewed before, we consider two kind of computing actions: training and aggregating. Let $\mathbf{RE}_v \in \mathbb{R}^{|\mathcal{V}^v|\times R}$ be the matrix representing the energy of using the resources per unit of time. Then, the consumed energy while using the computational resources $\psi\left(l_{model}, l_{\mathbf{w}}, \mathbf{tn}_v, \mathbf{A}_{G^v}, D_v^{(train)}, \mathbf{R}_v, \mathbf{RE}_v\right)$ within an administrative domain $v \in \mathcal{V}$ is

$$\psi\left(l_{model}, l_{\mathbf{w}}, \mathbf{tn}_v, \mathbf{A}_{G^v}, D_v^{(train)}, \mathbf{R}_v, \mathbf{RE}_v\right) \coloneqq \left\|l_{model}(\mathbf{tn}_v \circ D_v \circ time(\mathbf{R}_v))\mathbf{RE}_v\right\|_{1,1} + \left\|l_{\mathbf{w}}\left(\mathbf{1}_{|\mathcal{V}^v|}\mathbf{A}_{G^v} \circ time(\mathbf{R}_v)\right)\mathbf{RE}_v\right\|_{1,1},$$

where $time: \mathbb{R}^{|\mathcal{V}^v|\times R} \to \mathbb{R}^{1\times|\mathcal{V}^v|}$ is a function that maps the resources used by each node into the time they take to perform one operation, and $\mathbf{D}_v = \{d_n\} \in \mathbb{R}^{1\times|\mathcal{V}^v|}, d_n = \left|D_{v,n}^{(train)}\right| + \sum_{\overrightarrow{ni}\in\mathcal{E}^v,\ i\in\mathcal{V}^v}(1 - tn_i)\left|D_{v,i}^{(train)}\right|$, i.e., a vector with the data samples of the nodes. Here, $\mathbf{1}_k$ is a row vector of ones with k elements.

Once analysed all steps, one can formulate the optimization problem as

$$\min \sum_{v\in\mathcal{V}} \zeta\left(\mathbf{ee}_v, D_v^{(train)}\right) + \xi(\mathbf{A}_{G^v}, \mathbf{ME}_v, \mathbf{TB}_v) + \psi\left(l_{model}, l_{\mathbf{w}}, \mathbf{tn}_v, \mathbf{A}_{G^v}, D_v^{(train)}, \mathbf{R}_v, \mathbf{RE}_v\right) + \sum_{n\in\mathcal{N}^G(v)} a'_{v,n} te_{v,n} l_{\mathbf{w}}$$

As the problem is focused on minimizing the energy consumption, one needs to impose some accuracy α and latency β bounds to the model for ensuring a correct performance.

As shown previously, the weights of node v depends on those of its neighbours, hence, on its neighbours' datasets. The accuracy of the model increases, as data samples belonging to different distributions are added and as data samples matching the, possibly, time-evolving input data statistics are considered. Then, for an administrative domain $v \in \mathcal{V}$,

$$acc\left(D_v^{(train)}, \bigcup_{w \in \mathcal{N}^{G'}(v)} D_w^{(train)}, D_v^{(test)}, \mathbf{R}_v, \mathbf{B}_v\right) \geq \alpha.$$

The time expend at local training within the administrative domains depends on the movements of the data and the assigned resources, both network and computational,

$$acc\left(D_v^{(train)}, \mathbf{R}_v, \mathbf{B}_v\right) \geq \beta.$$

Notice that the resources of the nodes within the administrative domains cannot exceed their respective maximum, i.e.,

$$0 \leq r_{i,j} \leq r_{i,j}^{max}, \qquad \forall i,j \in \mathcal{V}^v, \qquad v \in \mathcal{V},$$

$$\sum_{k=1}^{R} r_{j,k} \geq 0, \qquad \forall \vec{ij} \in \mathcal{E}^v, \qquad i,j \in \mathcal{V}^v, \qquad v \in \mathcal{V},$$

$$\sum_{k=1}^{R} r_{i,k} \geq 0, \qquad \forall i \in \mathcal{V}^v, \qquad tn_i = 1, \qquad v \in \mathcal{V}.$$

Notice also that bandwidth is assigned only at $\mathcal{E}^v$ edges, then

$$0 \leq b_{i,j} \leq 1, \qquad \forall i,j \in \mathcal{V}^v, \qquad v \in \mathcal{V},$$

$$0 \geq b_{i,j} - a_{i,j}^{(v)} > -1, \qquad \forall i,j \in \mathcal{V}^v, \qquad v \in \mathcal{V}.$$

4.9.1.3 Secure AI/ML Model Distribution within 3GPP Domain

Many of today's AI applications were trained on data in a centralised location (e.g., the cloud) and often in an offline fashion. Allowing the training and in particular the inference of AI models to take place in a more decentralised and agile fashion, federate learning capabilities form the foundation for this ambition. PREDICT-6G inherits and implements this approach by enabling federated learning and inference through the definition of dedicated E2E and domain-specific Management Services (MSs), as described in D3.1 (PREDICT-6G/D3.1/4, 2023). For the set-up of an AI/ML service, Section 9.1 in D3.1 (PREDICT-6G/D3.1/9.1, 2023) provides a detailed message sequence chart depicting the required interactions among the involved MSs, which form the basis for the section herein.

A challenge that comes with decentralising tasks is to control the access to them to request them and to retrieve information or simply processing it. And the exchange of AI/ML models is no different. Focussing on a 3GPP domain interconnecting with the PREDICT-6G E2E management domain, the figure below illustrates a functional mapping of PREDICT-6G's MSs to 3GPP's in respect to working with AI/ML models. In the 3GPP domain, the Network Data Analytics Function (NWDAF) may contain the Model Training Logical Function (MTLF), as defined in (3GPP/TS23.501/2023). If so, the NWDAF offers two main AI/ML model services (3GPP/TS23.501/7.2.12, 2023):

1. Enable consumers to request and get ML model information.
2. Enable consumers to receive a notification when an ML model matching the subscription (and environmental) parameters becomes available.

An AI/ML Application Function (AF) is akin to a PREDICT-6G domain-specific MS and interacts with the E2E MS. The AI/ML AF then interacts with the technology domain, here 3GPP, where the NWDAF which may contain the MTLF as well as the Analytics Data Repository Function (ADRF). Optionally, the ADRF can also be deployed as a separate component.

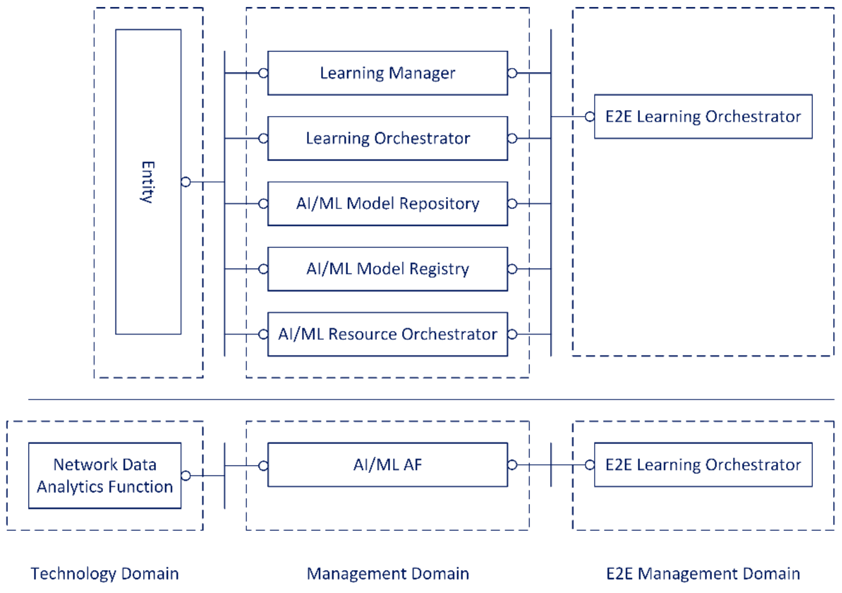


Figure 4-43. Mapping of PREDICT-6G Management Services to an AI/ML-Enabled 3GPP Technology Domain

Figure 4-44 below illustrates the interactions between PREDICT-6G's E2E and domain-specific Management Services with an AI/ML-enabled 3GPP domain. The sequence chart mimics the generic one provided in D3.1 (PREDICT-6G/D3.1/9.3, 2023). Most notably in that context is that the majority of orchestration and management in the domain-specific Management Services is a single AI/ML Application Function from 3GPP's view and the actual repositories for models and data(sets) is represented by the NWDAF. As illustrated in the chart below, the actual training of new models must be conducted outside of a 3GPP network and is not offered by an NWDAF or any other Network Function (NF).

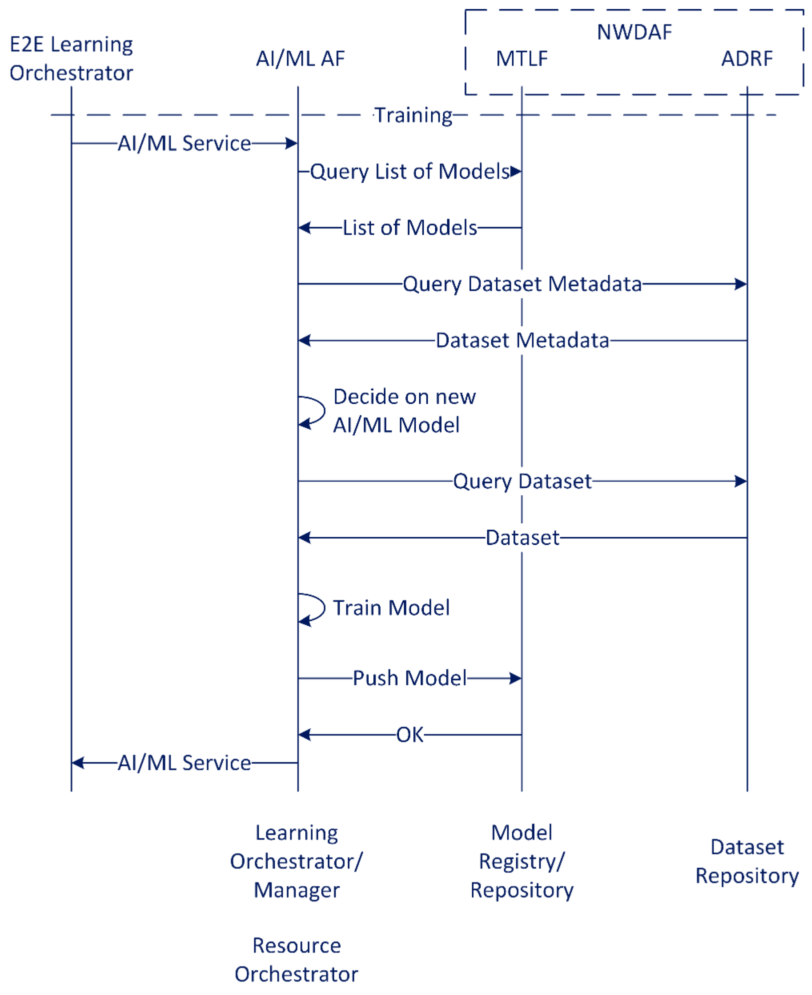


Figure 4-44. Message Sequence Chart for a AI/ML-Enabled 3GPP Domain Interacting with a PREDICT-6G Management Services

The **Figure 4-45** illustrates procedures the network enforced access control for AI/ML model distribution. An AI/ML model may be owned by a specific domain or entity, or a 3rd party (e.g., an application provider). It is important to ensure that only the intended parties can use and modify the AI/ML models, and to enforce a particular level of security based on security policies and contextual factors. Therefore, the procedure in **Figure 4-45** allows to ensure that AI/ML models are used or modified by the intended entities, and they are used/modified only under the intended conditions (e.g., satisfying environmental and contextual conditions). Examples of environmental or contextual information are PLMN ID, cell ID, serving area ID, time of day, date of the year, and weather conditions.

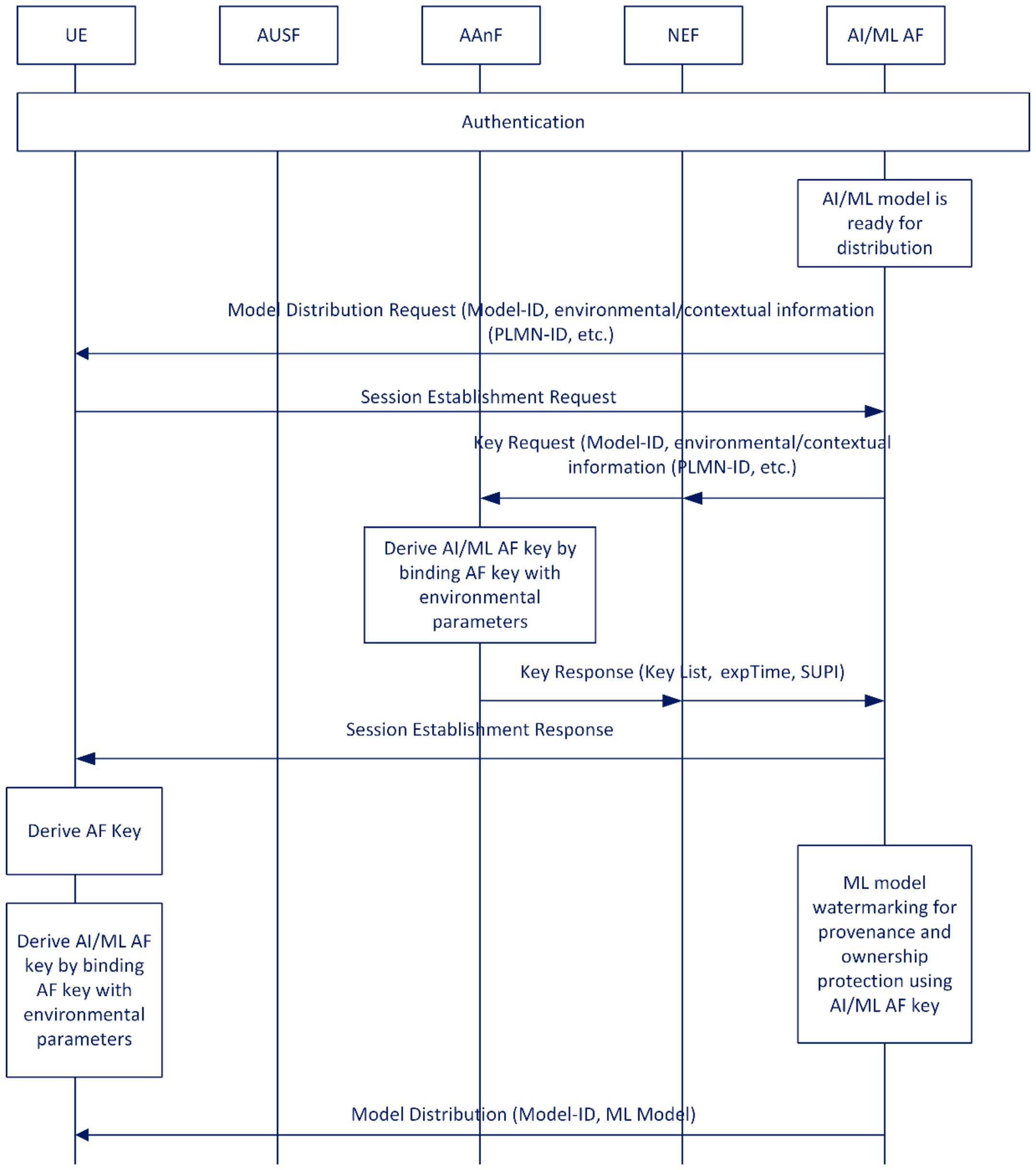

UE
AUSF
AAnF
NEF
AI/ML AF
Authentication
AI/ML model is ready for distribution
Model Distribution Request (Model-ID, environmental/contextual information (PLMN-ID, etc.)
Session Establishment Request
Key Request (Model-ID, environmental/contextual information (PLMN-ID, etc.)
Derive AI/ML AF key by binding AF key with environmental parameters
Key Response (Key List, expTime, SUPI)
Session Establishment Response
Derive AF Key
Derive AI/ML AF key by binding AF key with environmental parameters
ML model watermarking for provenance and ownership protection using AI/ML AF key
Model Distribution (Model-ID, ML Model)

Figure 4-45. Network Enforcement of AI/ML Model Distribution Access Control

4.9.2 Machine Learning-based Network Services

4.9.2.1 TWT settings in WiFi networks

Target Wake Time (TWT) is a feature of the IEEE 802.11ax-2021 standard that allows Wi-Fi stations to doze and wake up at predetermined times in order to minimize power consumption and channel contention. Even if it has originally been designed to save energy in low-power sensor networks, it can be used to obtain transmission delay determinism by avoiding the random backoff procedure. In fact, TWT wake intervals and durations can be set to limit the number of simultaneously active stations accessing the shared medium,

In D2.1, Section 4.4.1, we formalized a general traffic admission problem to guarantee given latency and throughput requirements to the requested traffic flows.

$$\max \sum_{f_{ab}} p_{ab} x_{ab}$$

$$l_{ab} x_{ab} \leq L_{ab}$$

$$T_{ab} \geq t_{ab} x_{ab}$$

More formally, given the set of Wi-Fi stations $S = \{s_1, \dots s_n\}$, we define the required deterministic traffic flows as $F = \{f_{11}, f_{12}, \dots f_{nm}\}$, For each traffic flow fab, the objective of the problem is to set the admission binary variable x_{ab} (which takes 1 if f_{ab} is admitted, 0 otherwise) that maximizes the sum of the priorities p_{ab} associated with each admitted flow. The latency and throughput performance l_{ab} and t_{ab} associated with the admission of the flow f_{ab} have to satisfy the flow requirement T_{ab} and L_{ab}.

The solution of the traffic admission problem requires to overcome several challenges:

- Wireless communication in general is very complex to model, as it is affected by many factors whose comprehensive impact is difficult to estimate, e.g., device mobility, radio interference, irregular traffic, etc. Without an accurate model, the behavior of the system cannot be predicted reliably.
- To guarantee reliable communication, not all settings can be applied in every conditions. Common machine learning approaches are not suitable to operate in mission-critical environments where a single bad decision could break determinism and therefore cannot be directly applied to solve this problem.

To overcome these challenges, we propose an approach based on constrained Reinforcement Learning (RL). In RL, agents learn to act in an unknown environment by trial and error, performing increasingly better actions as the learning phase progresses. Agents are not limited to specific behaviors; thus, they are free to explore the entirety of the action space as long as it leads to performance improvements. Instead, in constrained RL, the learning agent aims not only to optimize the total reward but also to satisfy additional constraints, which could be related to safety, budget, or diversity. In this case, those behaviors that are undesirable, as they may cause excessive performance loss or damages, are forbidden in all conditions, even during the initial exploratory phase. Clearly, constrained RL algorithms are more complex than existing techniques, as they have to address the challenges of learning dynamics, exploration, exploitation simultaneously with constraints satisfaction.

One of the most common approaches to solve a constrained RL problem is by using Lagrange multipliers, which are used to convert the constrained optimization problem to an unconstrained one with a new objective function. In the context of RL, the Lagrangian is a combination of the original reward function and the constraints, which are weighted by coefficients, known as Lagrange multipliers. Then, the optimal solution of the constrained problem is a stationary point of the Lagrangian, where the gradient of the Lagrangian is zero or parallel to the gradient of the constraints.

The Lagrange multipliers can be selected manually, which however leads to different optimal solutions that have to be evaluated in a time-consuming hyper-parameter tuning process. Alternatively, a primal-dual algorithm can be used to choose the parameters automatically. In this class of algorithms, the policy update is on a faster timescale than the multiplier update. Iteratively, the best policy for a given set of parameters is found, then the parameters are updated taking steps along the gradient of the Lagrangian with respect to the multipliers (Chen Tessler, 2018).

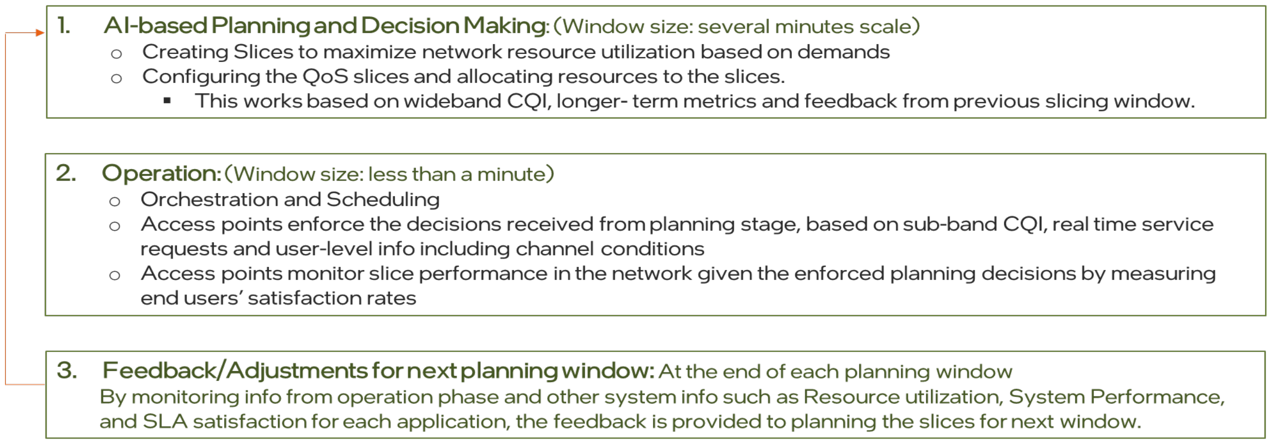
4.9.2.2 ML-driven radio access slicing in cellular networks

4.9.2.2.1 Network Slicing

In enterprise, it will be desirable to manage and optimize network operations to support multiple use cases with different quality of service (QoS) requirements. Network Slicing will allow multiple logical networks tailored to the different user requirements, over a common infrastructure. In this work our focus is on AI-based Dynamic Network Slicing. The goal is to allocate resources to assure service level agreements (SLA) and build intelligence to configure Radio Resources for Network Slices to guarantee:

i. Maximum delay for Low Latency (LL) cases
ii. Minimum data rate for high throughput (HT) cases
iii. while maximize radio resources available to other slices including best effort (BE)

The conceptual framework is shown in **Figure 4-46**. In this work, conventional round robin scheduling is used for operation. A constrained Reinforcement learning (RL) algorithm is designed for AI based slice planning.



1. **AI-based Planning and Decision Making:** (Window size: several minutes scale)
   - Creating Slices to maximize network resource utilization based on demands
   - Configuring the QoS slices and allocating resources to the slices.
     - This works based on wideband CQI, longer- term metrics and feedback from previous slicing window.

2. **Operation:** (Window size: less than a minute)
   - Orchestration and Scheduling
   - Access points enforce the decisions received from planning stage, based on sub-band CQI, real time service requests and user-level info including channel conditions
   - Access points monitor slice performance in the network given the enforced planning decisions by measuring end users' satisfaction rates

3. **Feedback/Adjustments for next planning window:** At the end of each planning window
   By monitoring info from operation phase and other system info such as Resource utilization, System Performance, and SLA satisfaction for each application, the feedback is provided to planning the slices for next window.

Figure 4-46. Conceptual Framework

4.9.2.2.2 Constrained RL Algorithm

The network slicing problem is formulated as a generic optimization problem:

$$\begin{aligned} \min_{\phi \in \Phi} \quad & \frac{1}{T_{\text{slice}}} \sum_{t=0}^{T_{\text{slice}}-1} f_0(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \phi)) \\ \text{s.t.} \quad & \frac{1}{T_{\text{slice}}} \sum_{t=0}^{T_{\text{slice}}-1} \mathbf{f}(\mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \phi), \mathbf{H}_t) \leq 0 \end{aligned}$$

Where, the objective f_0 is the (negated and normalized) mean-rate across best effort (BE) flows U_{be}. The constraints are defined to guarantee the average data rate of the worst performing user in H_T slice is greater or equal to r_{min}, and the 99-percentile of all the packet latencies observed during the slicing window for LL applications is less than l_{max}.

f = (f_1, f_2) is the vector-valued constraint function for min-rate (worst-rate) and max-latency (worst latency) constraints across high-throughput flows U_{ht} and low-latency flows U_{ll}, respectively. p(.) denotes the bandwidth allocation / slicing decisions.

$$f_0(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \phi)) = -\frac{1}{|\mathcal{U}_{be}(t)|} \sum_{i \in \mathcal{U}_{be}(t)} \frac{r_i(t)}{r_{\text{mean}}}$$

$$f_1(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \boldsymbol{\phi})) = \max_{i \in \mathcal{U}_{ht}(t)} \left(1 - \frac{r_i(t)}{r_{\min}}\right),$$
$$f_2(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \boldsymbol{\phi})) = \max_{i \in \mathcal{U}_{ll}(t)} \left(\frac{\ell_i(t)}{\ell_{\max}} - 1\right),$$

$r_i(t)$ denotes the rate/throughput of the flow *i* over the slicing window indexed with *t*. Similarly, $l_i(t)$ denotes the latency of the flow i over the slicing window *t*. Latency is measured as the 99-percentile of all the packet latencies observed during the slicing window *t*, where packet latency for each transmitted packet is calculated as the time difference between the transmission time and the generation time.

4.9.2.2.3 Lagrangian Duality and State-Augmentation

Given the vector of nonnegative dual multipliers corresponding to the constraints, we define the augmented Lagrangian:

$$\mathcal{L}_{\boldsymbol{\mu}}(\boldsymbol{\phi}) = \frac{1}{T_{\text{slice}}} \sum_{t=0}^{T_{\text{slice}}-1} \Big(f_0(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \boldsymbol{\phi})) + \boldsymbol{\mu}^T \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \boldsymbol{\phi}))\Big)$$

The optimal state-augmented parametrization minimizes the expected augmented Lagrangian w.r.t a dual multiplier distribution p**μ**:

$$\boldsymbol{\phi}^\star = \operatorname*{argmin}_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \mathbb{E}_{\boldsymbol{\mu} \sim p_{\boldsymbol{\mu}}} \mathcal{L}_{\boldsymbol{\mu}}(\boldsymbol{\phi}).$$

The offline training phase aims to solve the preceding minimization problem. During online-execution phase, we derive the slicing decisions using the trained state-augmented parametrization augmented with the current dual multipliers which evolve over time according to the dual dynamics given as

$$\boldsymbol{\mu}_{k+1} = \left[\boldsymbol{\mu}_k + \frac{\eta_{\boldsymbol{\mu}}}{T_0} \sum_{t=kT_0}^{(k+1)T_0=1} \mathbf{f}(\mathbf{p}(\mathbf{H}_t, \boldsymbol{\mu}_t; \boldsymbol{\phi}^\star), \mathbf{H}_t)\right]_+$$

4.9.2.2.4 Network Traffic Model

We consider three different slice/flow/user/traffic categories with different QoS requirements:

- High-throughput (HT): minimum rate/throughput constraint $r_{\min}$ (specified in bps/Hz).
- Low-latency (LL): maximum latency constraint $l_{\max}$ (specified in ms).

- Best-effort (BE): no strict QoS constraint. We target best possible rate/throughput for these traffics, given HT and LL traffics satisfy their respective requirements.

We fix the total number of flows N in each network but the fraction of flows in each category will vary from network to network. In our experiments, N = 20.

Traffic demands and packet generation for all flows follow the Constant Bit Rate Model (CBRM). Traffic demands for flows in each slice category vary over slicing windows in a random-walk like manner within prescribed ranges. CBRM simulates the packet arrivals for each flow based on the arrival rates dictated by the traffic demands.

4.9.2.2.5 Experiment Setup

Slicing decisions are made for each slicing window indexed with $t = 0, 1, \ldots, T_{slice}-1$, where T_{slice} is the total number of slicing windows over which the objective and the constraints are evaluated. Within each slicing window, multiple scheduling sub-windows take place.

In each scheduling sub-window, within each slice category, all flows take turns to transmit their packets via a round-robin scheduling algorithm. During transmission, the whole bandwidth allocated to the slice category is utilized. The main parameters of the experiment configuration are as follows:

- Total bandwidth BW = 20 MHz, transmit power is P_{tx} = 10 dBm, noise PSD is N_0 = -174 dBm/Hz.
- Total number of slicing windows T_{slice} = 200, dual multiplier and slicing decision update period T_0 = 5.
- The QoS requirements are set as r_{min} = 0.6 bps/Hz and l_{max} = 10 ms. The normalization factor in the mean best-effort rate objective is set to $r_{mean} = r_{min}$.
- For the neural network parametrization, we use a multi-layered perceptron (MLP) with two hidden layers with 64 neurons. As slice feature inputs, we concatenate the vector of fraction of flows in each slice and the average traffic demands.
- The primal learning rate (lr) is set as 0.001 and dual update step size as 2.
- The dual multipliers for each constraint are sampled uniformly in the range [0, 5].
- The training is done over 64 different training networks with a batch size of 16, over 25 epochs.
- We test on 16 different test networks and report the evolution of objective and constraint metrics.
- Two simple baselines of interest are:
    - Uniform slicing: Split the total bandwidth (BW) available equally among all slices (E.g., p = (⅓, ⅓, ⅓) for HT, LL and BE flows)

- Proportional slicing: Split the total BW available proportional to number of flows in each slice (E.g., if there are 10 H_T, 6 LL and 4 BE flows at a given time, the slicing decision is $p = (\frac{1}{2}, 0.3, \frac{1}{5})$).

4.9.2.2.6 Initial Simulation Results

The initial simulation results (code not open-sourced) show that the proposed state augmented algorithm outperforms both proportional and uniform slicing.

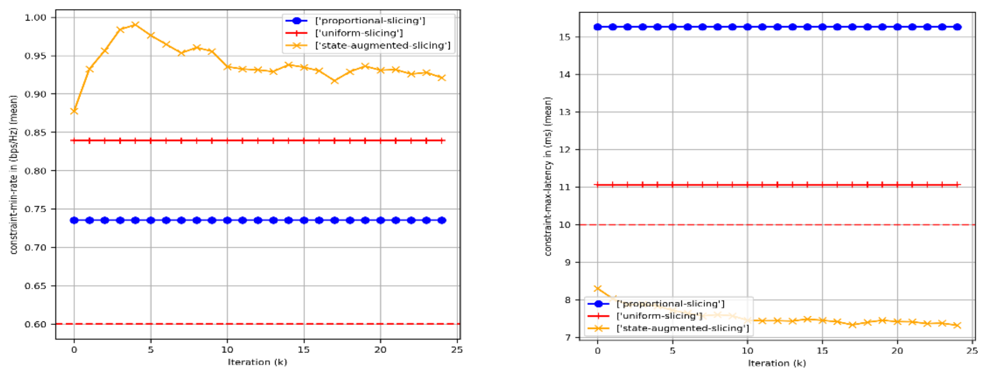


Figure 4-47. The test performance of the state-augmented model in terms of min-rate constraint and max-latency constraint for an average test network w.r.t. the number of training iterations. Rate (left) and Latency (Right) Constraints, Red dashed lines with no markers specify the QoS requirements. State-augmented slicing satisfies both QoS requirements for an average network unlike the two baselines.

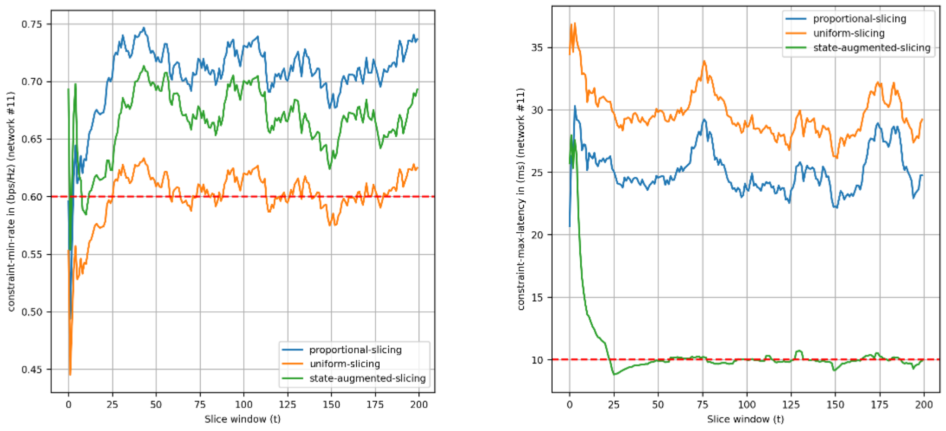


Figure 4-48. The time/slice evolution of the min-rate constraint (left) and max-latency constraint (right) for an example test network. For this network configuration, latency constraint is the harder of the two which is only met by the state-augmented approach.

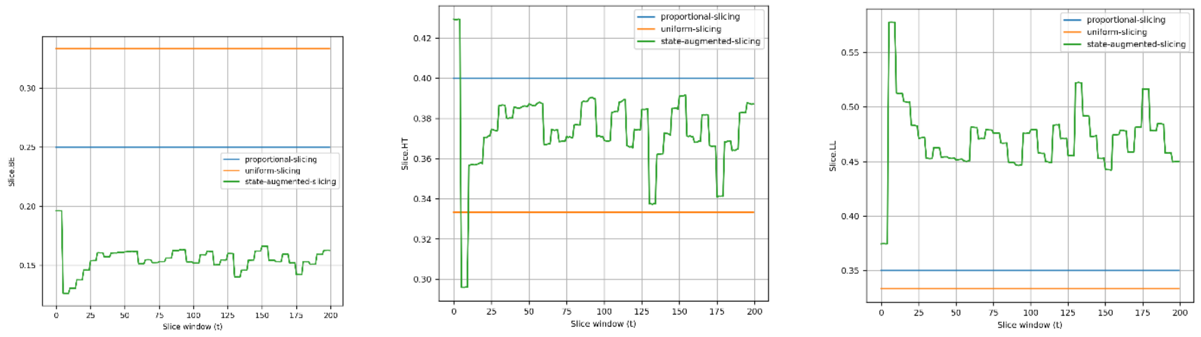


Figure 4-49. From left to right, the time/slice evolution of the bandwidth allocations for HT, LL and BE slices for a test network.

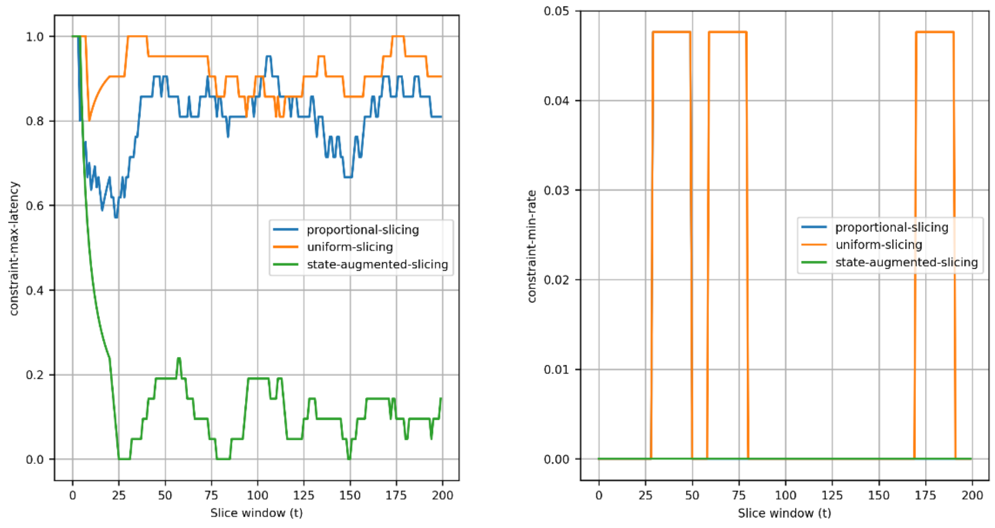


Figure 4-50. The average rate of failures for min-rate constraint (left) and max-latency constraint (right) for the same test network. The rate of failure metric captures what fraction of the time a typical user does not meet their respective QoS requirements.

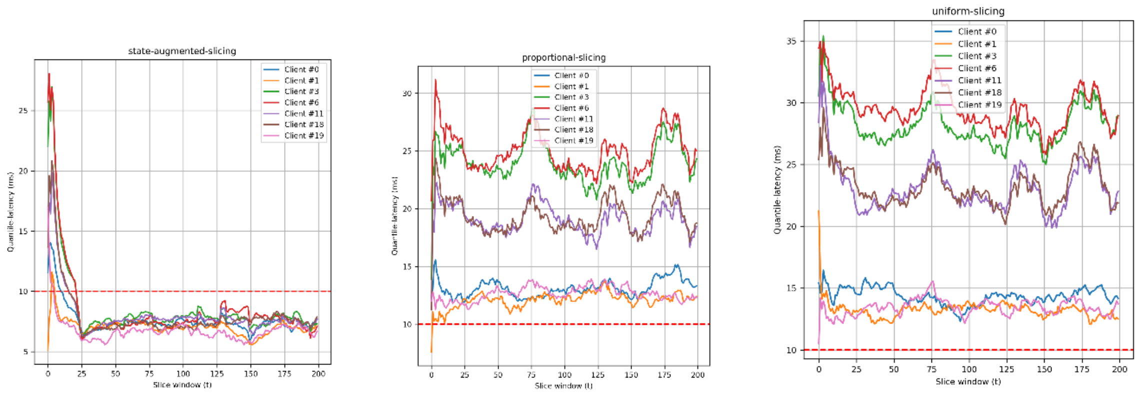


Figure 4-51. Ergodic average of latencies of LL flows over time, RL algorithm (Left), Proportional slicing (middle) and uniform slicing (right).

4.10 ML Architectural Framework and Interfaces

The concept of MLOps is defined in literature as the technology paradigm based on the extension of the DevOps methodology for enabling the full lifecycle management of AI/ML models in production environments. As part of this approach, MLOps targets the integration of different processes related to data collection, data transformation or model training towards the development and deployment AI/ML models (Erich et al. 2014).

Leveraging this technology concept, ATOS brings to PREDICT-6G the design of a specific AI/ML framework solution that targets the application of MLOps main principles towards the design of the future 6G systems, where AI/ML techniques are expected to play a significant role (A. Bang et al. 2023).

ATOS AI/ML framework is designed to support and facilitate the development and execution of AI/ML algorithms across multiple domains. The framework enables the management of the complete lifecycle of AI/ML models, from design and training to deployment in production environments at different domains, allowing their integration into the Management & Orchestration domain control loops. The integration of the AI/ML framework in these domains can be performed either through the exchange of AI/ML models in the shape of artifacts or through the delivery of AI/ML inference results, i.e., the resulting predictions or model outputs.

To host and deploy all the aforementioned functionalities in a portable and scalable manner, the AI/ML framework relies on a deployment strategy based on the concept of containerization and its orchestration through Kubernetes (K8s) system. K8s allows to scale and manage containerized software components and execute them as cloud-native applications. Based on this approach, ATOS relies on the use of Kubeflow cloud-native platform for enabling the deployment and orchestration of the AI/ML framework in any infrastructure where Kubernetes is available. Kubeflow[15] is an open-source system conceived for deploying and orchestrating ML workflows on K8s. It offers an ML toolkit where the SW Vendor workflow pipeline can be developed, executed, and tested. The development of workflows is enabled through Jupyter notebooks[16] servers and katib[17] components. With respect to orchestration, Kubeflow integrates an orchestration component called Kubeflow Pipelines.

[15] https://www.kubeflow.org/
[16] https://jupyter.org/
[17] https://github.com/kubeflow/katib

Funded by the European Union

Considering the described deployment framework, ATOS proposes an AI/ML framework architecture composed of five main blocks: Pipeline Development, Pipeline Orchestration Plataform (POP), Model Storage and Model Serving:

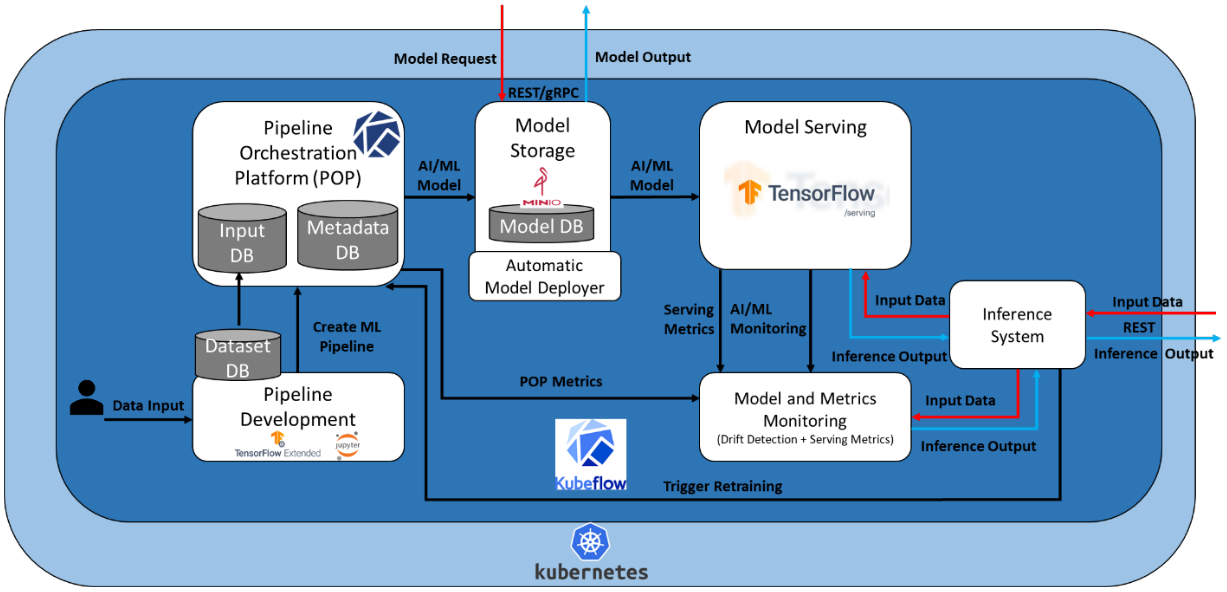


Figure 4-52. Current AI/ML framework architecture and technologies

Pipeline Development: The proposed AI/ML framework exploits the concept of AI/ML pipelines or workflows for managing the complete lifecycle of AI/ML models. This module is dedicated to developing AI/ML pipelines considering the combination of different data science and AI/ML stages such as data ingestion, data preparation, data validation, AI/ML model design, AI/ML model training, or AI/ML model testing, among others. This pipeline development module will run over Kubeflow through the preparation of Jupyter Notebooks, where TensorFlow[18] Extended (TFX) libraries are used. TFX is an extension of the well-known TensorFlow framework, which is typically used to create and manage ML production pipelines.

DatasetDB: Connected to the pipeline development inputs, a specific DB is included to allow the storage of input datasets. The type of database (relational, non-relational, supporting time-series) will be decided depending on the type of data to be stored.

Pipeline Orchestration Platform (POP): AI/ML framework module dedicated to performing and E2E orchestration and automatization of the AI/ML pipelines/workflows generation. The POP will not only build, manipulate, schedule, and deploy these pipelines in a portable and

[18] https://www.tensorflow.org/

scalable manner but will also orchestrate workflows using containerization strategies. The POP also includes the presence of two different databases: (i) the InputDB, used to store the data used as an input for performing the model training, and (ii) the MetadataDB, used to store metadata about the pipeline orchestration stages. Currently, the AI/ML framework relies on Kubeflow Pipelines[19] as POP while the DBs rely on MySQL and MinIO[20] respectively.

Model Storage: AI/ML framework module envisaged to store the ML models generated as the outcome of the different orchestrated pipelines. The model storage module has available a REST/gRPC interface to serve models under request. Currently, the model storage module of the AI/ML framework relies on MinIO database, which is merely a K8s-native objects storage.

Automatic Model Deployer: AI/ML framework add-on deployed on top of the Model Storage module to monitor the creation of new models. If new models are created, they are forwarded to the Model Serving module.

Model Serving: AI/ML pipelines/workflows are combined with model serving techniques for enabling the access to AI/ML inference results. This module offers the model serving capabilities. The current version of the AI/ML framework relies on TensorFlow Serving (TFS) for offering model serving. TensorFlow Serving is a flexible serving system where AI/ML models are deployed and are made available for inferencing requests through an API. The Model Serving module is also deployed over Kubeflow and directly connected to the Inference System and the Model and Metrics Monitoring module.

Inference System: AI/ML framework module devised to act as the main entry point for performing inference requests and returning inference outputs. This module is directly connected to the Model Serving for accessing the models, but also to the Model and Metrics Monitoring and to the POP for performing a closed-loop monitoring of the models and triggering its retraining in case of drift detection events.

Model and Metrics Monitoring: AI/ML framework module composed of two sub-modules dedicated to monitor the drift introduced by the AI/ML models and to monitor specific metrics related to Model Serving performance. The module is connected to the Model Serving for monitoring models and receiving serving metrics, but also to the inference system for informing about drift detection, and to the POP for receiving POP performance metrics.

[19] https://www.kubeflow.org/docs/components/pipelines/v1/introduction/
[20] https://min.io/

4.10.1 SW Design

The AI/ML Framework described in the previous subsection will include some functionalities present in D3.1 (PREDICT-6G/D3.1, 2023) united in a single module. Particularly, the AI/ML Framework will cover functionalities considered in the following MS:

- AI/ML Model Registry (Fully covered): Store and serve specific information and characteristics related to the models already onboarded in PREDICT-6G platform. This functionality is already available in the framework.
- Dataset Registry (Fully covered): Store and serve specific information related to datasets available for model training or pre-training. This functionality is not available in the framework, but it will be covered as part of the framework extension plan.
- Model Repository (Fully covered): Store and serve the AI/ML models already onboarded in the platform through the Learning Manager. This functionality is already available in the framework.
- Dataset Repository (Fully covered): Allow the storage and serving of the datasets already onboarded in the platform through the Learning Manager. This functionality is not available in the framework, but it will be covered as part of the framework extension plan.
- Resource Orchestration/Learning Manager (Partially covered): Allow the creation of new AI/ML models, allow the monitoring and update and redeployment of AI/ML models according to specific metrics or external requests. This functionality is already available in the framework.

Considering all previous points regarding the MS functionalities that the AI/ML framework will cover, the following subsection provides details of the interactions that will be performed between the AI/ML framework module, and the rest of MS/modules present in the architecture. All interactions are based in the Diagram present in Section 9 of the 3.1 Deliverable [PREDICT-6G/D3.1/9, 2023], assuming that the AI/ML framework will cover the role of the MS specified at the beginning of the section.

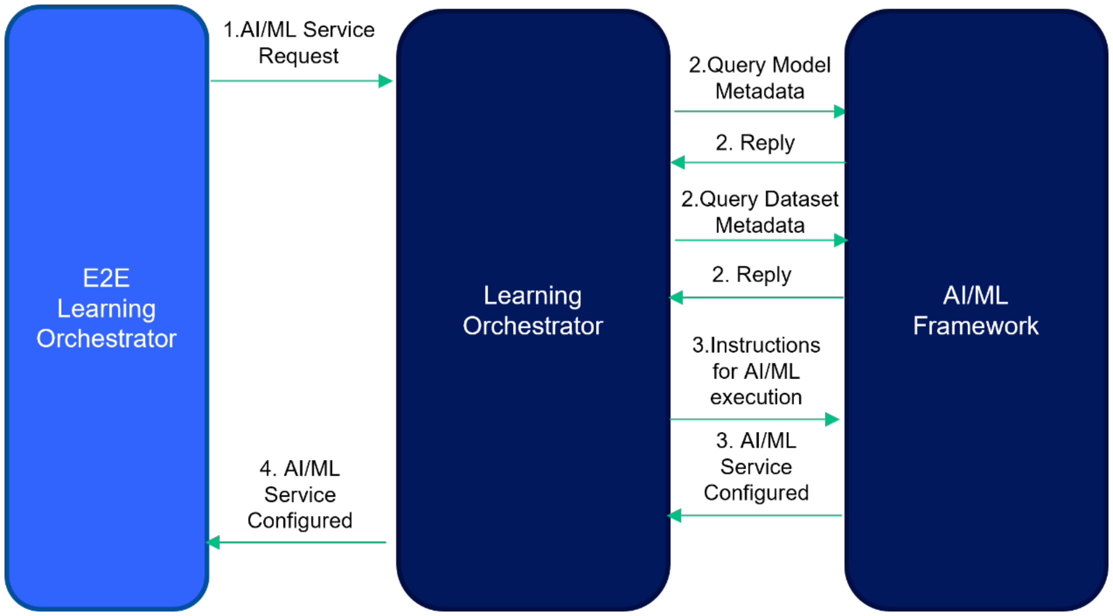


Figure 4-53. AI/ML Framework interactions

Figure 4-53 depicts the workflow of the interactions between AI/ML framework module and the Learning Orchestrator module at the MD level. Particularly, this workflow depicts the setup that takes place when a new AI/ML Service is requested from the E2E Learning Orchestrator, enabling its implementation at the corresponding domain. In this regard, the performed interaction between these components will be the following:

1. The E2E Learning Orchestrator sends a new AI/ML Service request generated from a user to set up the configuration of a new E2E Service. The E2E learning Orchestrator will forward the request to the corresponding domains involved in the configuration towards the MD Learning Orchestrator. Since this is a unidirectional communication sending information towards the module, its implementation will be performed using a REST interface.
2. The MD Learning Orchestrator receives the request and processes it. With the information included in its content, the MD Learning Orchestrator proceeds to retrieve from the AI/ML framework module the metadata of the available models in order to select the most appropriate model for the AI/ML operations. Similarly, the Learning Orchestrator also retrieves the necessary datasets metadata for training the AI/ML model. Both messages will be implemented using a REST interface.
3. With all the information compiled in the previous queries, after determining the most appropriate algorithm to run within the domain, the Learning Orchestrator provides the

AI/ML framework with the instructions to run the chosen AI/ML tasks involved in the configuration of the E2E service. After the service was configured within the domain, the AI/ML module replies with a confirmation message. This interaction follows a standard REST model since it is a simple confirmation message.

4. Finally, the MD Learning Orchestrator forwards the reply to the E2E Learning Orchestrator to confirm the configuration of the E2E service within the domain. In the same fashion, this confirmation message will also use a REST implementation.

Considering the expected interactions and functionalities described in this section, the project will reuse, adapt and extend the AI/ML framework for developing and deploying AI/ML algorithms developed in T3.1. To perform this extension, the following framework enhancements will be implemented:

1. **Add Dataset Registry and Repository functionalities:** PREDICT-6G requires the inclusion of dataset registry and repository capabilities for allowing the storage of complete data sets as well as their most relevant characteristics. Since the current version of the AI/ML framework only supports data ingestion without exploiting storage capabilities, the project will extend the framework functionalities to enable dataset registry and repository functionalities. This extension will be focused on the development of a specific REST interface for accessing dataset repository and registry information stored in DatasetDB.
2. **Migrate AI/ML framework from TensorFlow to Pytorch libraries:** PREDICT-6G partners plan to develop AI/ML algorithms under Pytorch[21] framework. As a consequence, the existing AI/ML framework will be adapted to support Pytorch, enabling the pipeline generation, orchestration, model storage and model serving for Pytorch libraries.
3. **Extend AI/ML framework capabilities to support Federated Learning:** The AI/ML models developed in T3.1 will exploit Federated Learning techniques. The AI/ML framework will be shaped to support FL models where training and evaluation phases are clearly split.
4. **Extend AI/ML framework capabilities to support the development and deployment of Distributed Learning and, potentially Reinforcement Learning models:** The AI/ML models developed in T3.1 will also leverage DL and RL techniques. The AI/ML framework will be extended to support these models.
5. **Integrate AI/ML framework with the rest of AI/ML Management Services:** The project will integrate the AI/ML framework with the rest of AICP components through the use of the

[21] https://pytorch.org/

existing REST API/gRPC interfaces exposed in the Model Serving and Model Storage components.

Considering the targeted extensions, the SW design of the AI/ML framework will be slightly modified through the modification of some specific submodules. The set of modifications or additions are described below:

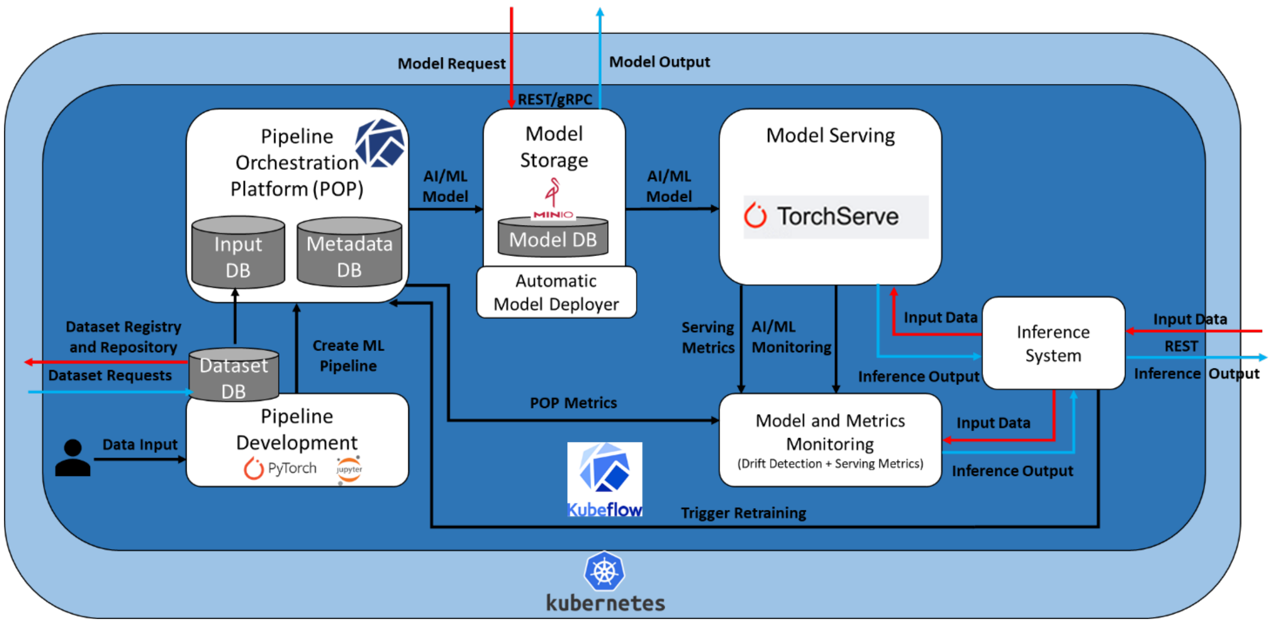


Figure 4-54. Proposed AI/ML framework architecture

Pipeline Development: This module will be extended to support the development of Pytorch E2E production pipelines through the potential use of Pytorch libraries such as TorchX or ZenML. This pipeline development will be enhanced to support the distribution of data from several data sources in the context of Federated Learning and, potentially, Reinforcement Learning techniques.

Pipeline Orchestration Platform (POP): This module will be extended to support the development of pipelines for Federated Learning through the replication of specific pipeline stages related to data validation, preparation, etc. The module will still rely on Kubeflow Pipelines since this framework also supports Pytorch pipelines.

Model Serving: This module will also be extended to support the serving of Pytorch AI/ML models through the potential use of Pytorch libraries such as Torchserve. As for the pipeline development, the serving design will be performed to support the distribution of FL/RL models.

DatasetDB: As previously mentioned, a specific interface will be developed to access dataset registry and repository functionalities allowing to retrieve complete data sets as well as their most relevant features. The type of database (relational, non-relational, supporting time-series) will be decided depending on the type of data to be stored.

4.10.2 Early implementation and Release Plan

The existing implementation of the AI/ML framework is available in Atos private repository:

https://github.gsissc.myatos.net/GLB-BDS-ETSN-GLB/MLOPS-FRAMEWORK.

To upgrade and enhance the capabilities of the existing version, an implementation plan is defined, consisting initially of four SW development phases, concluding at M15, M18, M21 and M24, according to the roadmap reported in Section 5. The first phase is expected to cover the functionalities related to adding dataset registry and repository capabilities, as well as the support of Pytorch libraries. The second phase will add functionalities to support the deployment of FL algorithms. The third phase will include the fine-tuning of the previous functionalities and the release of the first complete version of the AI/ML framework. Finally, the last phase at M24 will deliver the final version of the AI/ML framework, covering a complete integration with other MS.

Regarding frameworks, programming languages and libraries, this module will be enhanced by exploiting Pytorch framework and Python as the main programming language. Regarding libraries, Pytorch libraries like TorchX, TorchServe or ZenML are expected to be used. With respect to the DB, it could be developed by using PostgreSQL[22], SQLite[23], InfluxDB[24] or other types of databases depending on the type of data to be stored.

[22] https://www.postgresql.org/
[23] https://www.sqlite.org/index.html
[24] https://www.influxdata.com/

5 Implementation Roadmap

This section shows the roadmap towards the release of an AICP prototype, following the time management methodology discussed in Section 3.3. The roadmap is exposed in **Figure 5-1** and considers a timeframe that spans from M13 to M27, when WP3 activities will be terminated, encompassing 4 development cycles plus a final one for the prototype refinement and reporting (D3.3 and D3.4)

It is important to note that as soon as feedback from the implementation activities are collected, the releases of the different AICP components might be updated, especially for those modules that directly interacts with the data plane, whose implementation depends on the interfaces exposed by the MDP. Integration AICP-MDP integration activities are foreseen, according to the definition of Development cycle: those activities are carried out in the context of WP4 and are not explicitly reported in the roadmap figure.

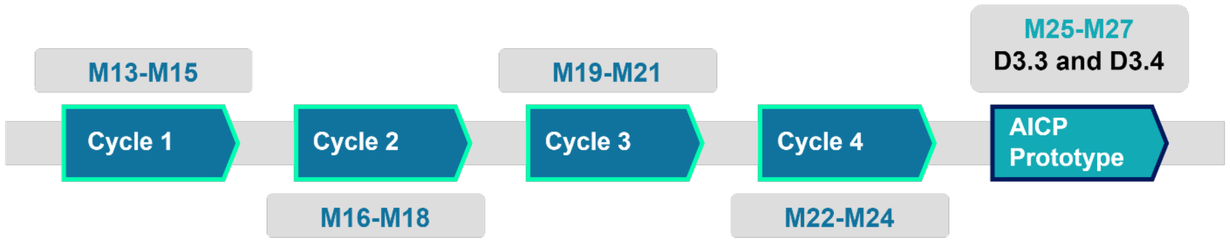


Figure 5-1. Initial roadmap towards AICP prototype

In **Table 5-1** are reported the advancements expected in the implementation of the different AICP components.

Table 5-1. Implementation advancements expected at the end of each Development Cycle

Component	Cycle 1 M13-M15	Cycle 2 M16-M18	Cycle 3 M19-M21	Cycle 4 M22-M24
Time Sync	Initial interfaces and data models definition	Implementation of technology domain specific TS MS	Implementation of E2E time sync management	Integration with MDP
Service Ingestion	Component skeleton with base NBI functions and	NBI for service Management, service management	Base authentication logic design and early	Full-fledged NBI and SBI. Integration with Service

Component	**Cycle 1** **M13-M15**	**Cycle 2** **M16-M18**	**Cycle 3** **M19-M21**	**Cycle 4** **M22-M24**
	base service parsing logic	logic, initial integration with service automation via SBI	implementation. Possible changes in the information model to be integrated	Automation and IAM platform under refinement
Exposure Services	Skeleton of MD level for topology exposure service. Data model analysis for mock modules	Mock for MD topologies.	Skeleton form E2E MD level modules. Mock to complete modules, if possible	Complete set of exposure modules in both MDs and E2E Md levels
Service Automation	Final definition of data model. Development of MD Service Automation and interfaces.	Implementation of E2E Service Automation functionalities and interfaces.	Fine-tuning MD and E2E functionalities, initial integration with the rest of modules.	Final integration and testing with other modules.
Path Computation	Initial interfaces and data models definition	Interfaces and data models consolidation Preliminary path computation engine	Consolidated Path Computation engine both at domain and E2E level Initial integration	Fully integration of the Path Computation module with the other modules
Digital Twin	Initial interfaces and data models definition	Preliminary Queuing models for deterministic services	KPIs composition for e2e estimation	Fully Integration with other models

Component	**Cycle 1** **M13-M15**	**Cycle 2** **M16-M18**	**Cycle 3** **M19-M21**	**Cycle 4** **M22-M24**
Resource Configuration	Initial manually configurable TSN module	Preliminary TSN configurator with DetNet Constraints focus	Integrable TSN configurator in DetNet with well-defined SBI endpoints	Fully autonomous interoperable configurator
Data Collection and Management	Integration with MDP available data collection technologies	Support for hierarchical deployment, Integration with MDP available data collection technologies	Support of IAM, integration with MDP available data collection technologies	Integration with MDP available data collection technologies
AI/ML algorithmic framework	Problem Formulation and initial algorithm definition for distributed learning	Initial Performance evaluation through numerical analysis	Consolidated version of the algorithmic frameworks	Integration in the AI/ML learning task operational flow
ML Architectural Framework	Addition of dataset registry and repository capabilities. Support for Pytorch libraries.	Support for the deployment of Federated Learning algorithms.	Fine-tuning of functionalities, and release of first complete version of AI/ML architectural framework. Initial integration with other Management Services.	Final integration with the rest of Management Services.

6 Conclusions

This document aims at presenting the initial advancements related to the implementation of PREDICT-6G AICP. Starting from the control plane functional architecture presented in D3.1, a methodology to select MSs candidate for implementation has been presented, along with a strategy to manage the time dedicated to the development and integration activities and consideration related to the source code management and influence of the MDP technology on the implementation of some MSs. The MSs selected have been grouped per SW components, whose SW design and implementation plan have been also described. Finally, a roadmap mapping the releases of the different piece of SW have been reported.

Nevertheless, the work presented in the deliverable is only a starting point towards the realisation of an AICP prototype, ready to be integrated with the MDP prototype, exploiting the activities in WP4. The path is not straightforward: the implementation activities often reveal issues and gaps not identified at the design phase. For that reason, the time remaining to the end of the WP3 has been fractionated in implementation cycles, at the end of which, beside the SW artifacts, key feedback from implementation and integration are collected and used to update the AICP functional and SW design (whose final version will be reported in D3.3 and D3.4, respectively), progressively and up to the prototype release.

7 References

(PREDICT-6G/D1.1/4-5, 2023) PREDICT-6G D1.1 "Analysis of use cases and system requirements.", Sections 4 and 5, Jun. 2023.

(PREDICT-6G/D1.2/5.3.3, 2023) "PREDICT-6G framework architecture and initial specification", Section 5.3.3, Dec. 2023

(PREDICT-6G/D1.2/5.3.3.1, 2023) "PREDICT-6G framework architecture and initial specification", Section 5.3.3.1, Dec. 2023

(PREDICT-6G/D1.2/5.4.3, 2023) "PREDICT-6G framework architecture and initial specification", Section 5.4.3, Dec. 2023

(PREDICT-6G/D3.1/2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Sep. 2023

(PREDICT-6G/D3.1/4, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 4, Sep. 2023

(PREDICT-6G/D3.1/6.2, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 6.2, Sep. 2023

(PREDICT-6G/D3.1/6.2.1.2-6.2.2.2, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Sections 6.2.1.2 and 6.2.2.2, Sep. 2023

(PREDICT-6G/D3.1/7.2, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 7.2, Sep. 2023

(PREDICT-6G/D3.1/7.2.1.1-7.2.2.1, 2023) PREDICT-6G D3.1 "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Sections 7.2.1.1 and 7.2.2.1, Sep. 2023

(PREDICT-6G/D3.1/7.2.1.3-7.2.1.5-7.2.1.6-7.2.1.7-7.2.2.6-7.2.2.7, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Sections 7.2.1.3, 7.2.1.5, 7.2.1.6, 7.2.1.7, 7.2.2.6 and 7.2.2.7, Sep. 2023

(PREDICT-6G/D3.1/7.2.1.4-7.2.2.4, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Sections 7.2.1.4 and 7.2.2.4, Sep. 2023

(PREDICT-6G/D3.1/9, 2023) PREDICT-6G D3.1 "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 9, Sep. 2023

(PREDICT-6G/D3.1/9.1, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 9.1, Sep. 2023

(PREDICT-6G/D3.1/9.3, 2023) "Release 1 of AI-driven inter-domain network control, management, and orchestration innovations.", Section 9.3, Sep. 2023

(3GPP/TS23.501/2023) "System architecture for the 5G System (5GS) v18.3.0", Sep. 2023

(3GPP/TS23.501/7.2.12, 2023) "System architecture for the 5G System (5GS) v18.3.0", Section 7.2.12, Sep. 2023

Malandrino, F. et al. (2023), 'Tuning DNN Model Compression to Resource and Data Availability in Cooperative Training,' in IEEE/ACM Transactions on Networking, doi: 10.1109/TNET.2023.3323023.

Gou, J., et al. (2021) 'Knowledge distillation: A survey,' International Journal of Computer Vision.

Gao, Z., et al. (2021), 'Knowru: Knowledge reusing via knowledge distillation in multi-agent reinforcement learning,' arXiv [cs.CR]. Available at: http://arxiv.org/abs/2103.14891

Zhang, T., et al., 'Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation,' arXiv [cs.CR]. Available at: http://arxiv.org/abs/2109.00525

Chen Tessler, et al. (2018). 'Reward constrained policy optimization,' arXiv [cs.CR]. Available at: http://arxiv.org/abs/1805.11074

L. Yang, Y. Lu, J. Cao, J. Huang and M. Zhang, "E-Tree Learning: A Novel Decentralized Model Learning Framework for Edge AI," in IEEE Internet of Things Journal, vol. 8, no. 14, pp. 11290-11304, 15 July15, 2021, doi: 10.1109/JIOT.2021.3052195.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. Advances in neural information processing systems, 30.

J. J. Alcaraz et al. (2022), "Model-Based Reinforcement Learning with Kernels for Resource Allocation in RAN Slices," IEEE Trans. Wireless Communications, 2022.

Navid NaderiAlizadehet al. (2022), "State-augmented learnable algorithms for resource management in wireless networks," IEEE Transactions on Signal Processing, 2022.

Y. Liu, et al. (2020), "A Constrained Reinforcement Learning Based Approach for Network Slicing," IEEE 28th Int'l. Conf. Network Protocols, 2020.

Q. Liu, et al. (2021), "OnSlicing: Online End-to-End Network Slicing with Reinforcement Learning," Proc. 17th Int'l. Conf. Emerging Networking Experiments and Technologies, 2021, pp. 141—53

M. Ruiz, D. Careglio, and L. Velasco, "CURSA-SQ models for Time-Sensitive Networking," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2023.

Erich F., Amrit C., Daneva M., “A Mapping Study on Cooperation between Information System Development and Operations”, In: Jedlitschka A., Kuvaja P., Kuhrmann M., Männistö T., Münch J., Raatikainen M. (eds) Product-Focused Software Process Improvement. PROFES 2014. Lecture Notes in Computer Science, vol 8892. Springer, Cham, 2014.

Ankur Bang, Kapil Kant Kamal, Padmaja Joshi, Kavita Bhatia, 6G: The Next Giant Leap for AI and ML, Procedia Computer Science, Volume 218, 2023, Pages 310-317, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.01.013.

8 Appendixes

8.1 Appendix A - AICP Operational Workflows

8.1.1 E2E Deterministic Service Provisioning

The E2E Service provisioning operational workflow describes the interactions between multiple MSs involved in the E2E and technology-specific MDs. The workflow is first described and illustrated in **Figure *8-1*** from the E2E perspective, and then, analysed step by step, including specific workflows for loops in local MDs (**Figure 8-2** and **Figure 8-3**). The AI/ML MSs are here indicated as AI/ML AI-based & Predictive Decision Service (E2E and not E2E) in a single box per domain, in order to ease the readability of the figures.

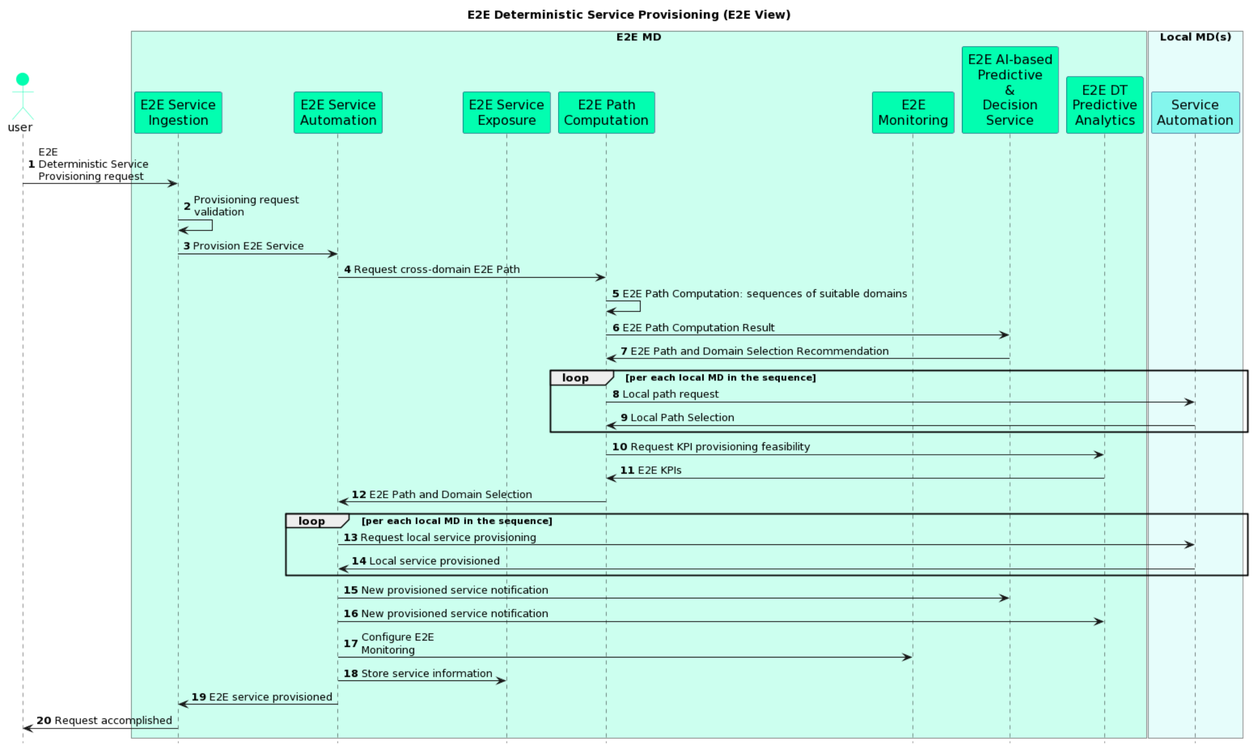

E2E Deterministic Service Provisioning (E2E View)
E2E MD
Local MD(s)
user
E2E Service Ingestion
E2E Service Automation
E2E Service Exposure
E2E Path Computation
E2E Monitoring
E2E AI-based Predictive & Decision Service
E2E DT Predictive Analytics
Service Automation
1 E2E Deterministic Service Provisioning request
2 Provisioning request validation
3 Provision E2E Service
4 Request cross-domain E2E Path
5 E2E Path Computation: sequences of suitable domains
6 E2E Path Computation Result
7 E2E Path and Domain Selection Recommendation
loop [per each local MD in the sequence]
8 Local path request
9 Local Path Selection
10 Request KPI provisioning feasibility
11 E2E KPIs
12 E2E Path and Domain Selection
loop [per each local MD in the sequence]
13 Request local service provisioning
14 Local service provisioned
15 New provisioned service notification
16 New provisioned service notification
17 Configure E2E Monitoring
18 Store service information
19 E2E service provisioned
20 Request accomplished

Figure 8-1 E2E Deterministic Service provisioning - E2E Management Domain view

- **Step 1.** *User/Operator* sends a Service Provisioning Request to the *E2E Service Ingestion MS.*

- **Step 2.** *E2E Service Ingestion* validates the format and the syntax of the Service Provisioning Request.
- **Step 3.** *E2E Service Ingestion* forwards the Service Provisioning Request to the *E2E Service Automation* for launching the provisioning process**.**
- **Step 4.** *E2E Service Automation* kicks off the provisioning process by requesting the cross-domain E2E path to the *E2E Path Computation.*
- **Step 5.** *E2E Path Computation* computes gross-grained E2E paths where the nodes in the network graph have the granularity of a domain, interconnected by the existing links between domain's border nodes.
- **Step 6.** *E2E Path Computation* sends the results to the *E2E Learning Orchestrator* for their evaluation.
- **Step 7.** *E2E Learning Orchestrator* recommends the best path and domain selection for the E2E, considering the path computation alternatives, and forwards this information to the *E2E Path Computation*.
- **Step 8** and **9.** After preselecting the domain and the path, the *E2E Path Computation* requests an E2E KPI computation to the *E2E DT Predictive Analytics MS, based in the KPIs computed for each local domain*.
- **Step 8.** *E2E Path Computation* starts an iterative path computation process in each local Management Domain by interacting with the corresponding technology-specific *Service Automation MS*. See details in **Figure 8-2**.
- **Step 9.** *E2E Path Computation* receives the local path selection from the different technology domains.
- **Step 10** and **11.** After preselecting the domain and the path, the *E2E Path Computation* requests an E2E KPI computation to the *E2E DT Predictive Analytics MS, based in the KPIs computed for each local domain*.
- **Step 12.** *E2E Path Computation* forwards the *E2E Path Computation* result to the *E2E Service Automation*.
- **Step 13.** *E2E Service Automation* triggers an iterative process for requesting the service provisioning in the corresponding domain through an interaction with the *Service Automation MS*.
- **Step 14.** *E2E Service Automation* receives feedback about the local service provisioning from the *Service Automation MS.*
- **Step 15.** *E2E Service Automation* sends a notification about the E2E service provisioning to the *E2E Learning Orchestrator.*
- **Step 16.** *E2E Service Automation* sends a notification about the E2E service provisioning to the *E2E DT Predictive Analytics MS***.**

Funded by the European Union

- **Step 17.** *E2E Service Automation* configures *E2E Monitoring* to collect from *Measurement Collection MSs* of each domain.
- **Step 18.** *E2E Service Automation* stores the E2E Service Information in the *E2E Service Exposure.*
- **Step 19.** *E2E Service Automation* informs about the *E2E Service Provisioning* to the *E2E Service Ingestion.*
- **Step 20.** *E2E Service Ingestion* informs the ***user or the operator*** about the success of the Service Provisioning Request.

In the following, we detail the first loop, i.e., the operations happening in local domains when the E2E Path Computation requests a local deterministic path.

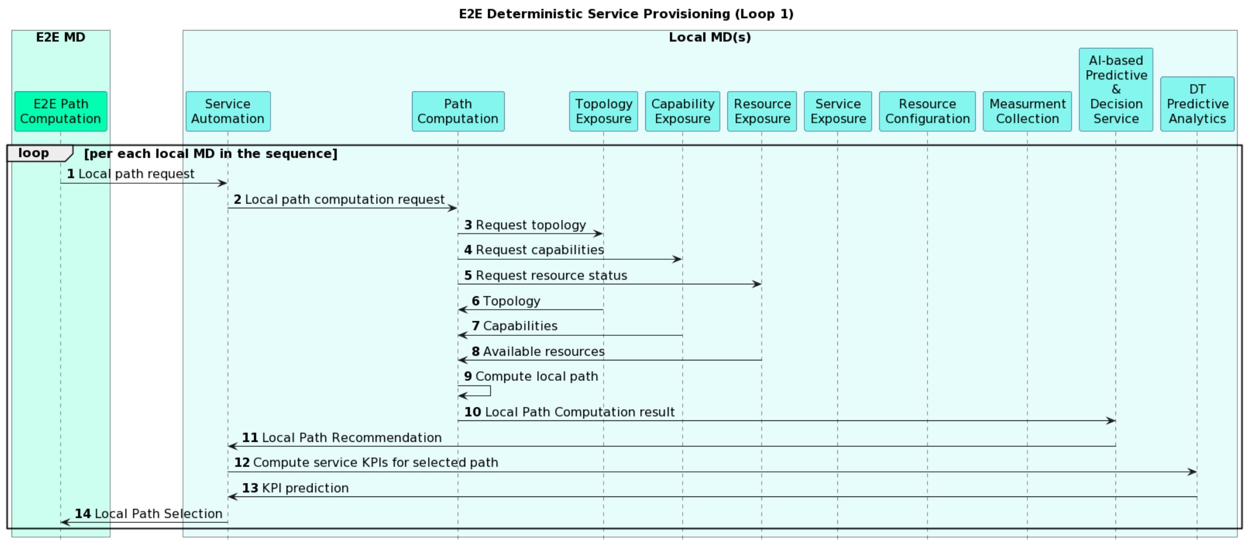


Figure 8-2 E2E Deterministic Service provisioning – Loop 1, Local Management Domain view

- **Steps 1** and **2.** *Service Automation MS* forwards the local path computation request to the local *Path Computation MS.*
- **Steps 3, 4** and **5.** *Path Computation MS* in each domain asks *Topology Exposure*, *Capability Exposure*, and *Resource Exposure* to retrieve information from domains.
- **Steps 6, 7,** and **8.** *Topology Exposure, Capability Exposure,* and *Resource Exposure* get the corresponding information from the data plane, abstract it, and send it back to *Path Computation.*

- **Steps 9, 10,** and **11.** *Path Computation* performs the path calculation for each domain and forwards this calculation to the *Learning Orchestrator* to get a recommendation for the best path alternative to provision the service in the specific local domain.
- **Step 12.** *Path Computation* forwards the selected path to the *Service Automation*
- **Steps 13** and **14.** Under request from *Service Automation,* the *DT Predictive Analytics* MS composes a dedicated scenario with the sub-topology defined for the route to evaluate, selects the already provisioned services that are supported by these resources and runs simulations to estimate the KPIs based on the definition of traffic for the new service and the traffic models for the current services. The estimation results for the local domain are returned.
- **Step 14.** *Service Automation* forwards the local path selection to the E2E Path Computation.

Below is detailed the second loop characterizing the provisioning of an E2E Deterministic service, i.e., the set of operations required to provisioning a local deterministic sub-service.

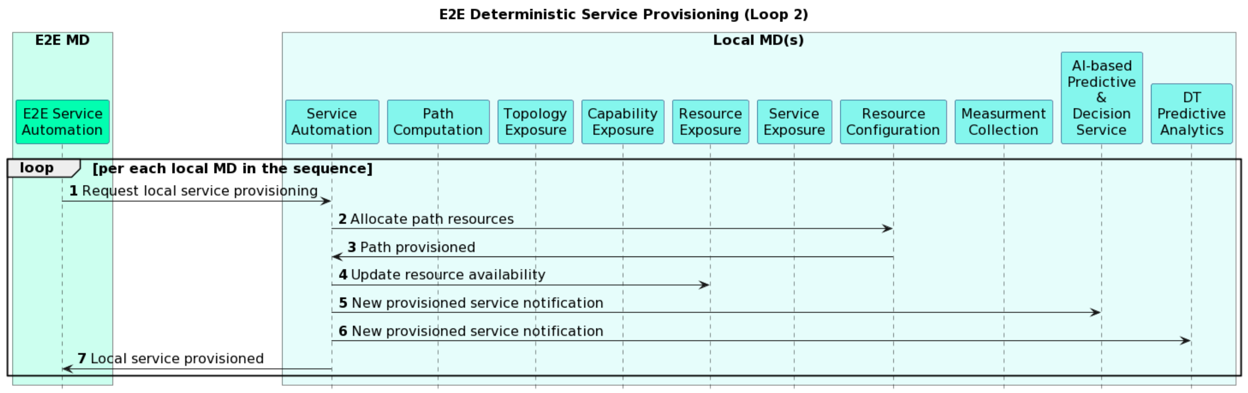

E2E Deterministic Service Provisioning (Loop 2)
E2E MD
Local MD(s)
E2E Service Automation
Service Automation
Path Computation
Topology Exposure
Capability Exposure
Resource Exposure
Service Exposure
Resource Configuration
Measurment Collection
AI-based Predictive & Decision Service
DT Predictive Analytics
loop
[per each local MD in the sequence]
1 Request local service provisioning
2 Allocate path resources
3 Path provisioned
4 Update resource availability
5 New provisioned service notification
6 New provisioned service notification
7 Local service provisioned

Figure 8-3 E2E Deterministic Service provisioning - Loop 2, Local Management Domain view

- **Step 1.** *E2E Service Automation* requests the service provisioning in the selected local domain through an interaction with the *Service Automation MS.*
- **Step 2.** *Service Automation MS* performs a request for resource configuration to the *Resource Configuration MS.*
- **Step 3.** *Resource Configuration MS* receives the request and configures resources directly on the data plane and provisions the path to the *Service Automation MS.*
- **Step 4.** *Service Automation MS* updates resource availability at the *Resource Exposure MS.*
- **Steps 5** and **6.** *Service Automation MS* informs the *Learning Orchestrator* about the service provisioning.

Funded by the European Union

- **Step 7.** *Service Automation MS* informs the *E2E Service Automation* about the service provisioning in the local domain.

8.1.1 E2E Deterministic Service Decommissioning

The termination of a deterministic service can happen under two potential situations: (1) explicit termination request from the user/operator, or (2) due to the expiration of the service lifetime, which is a subset of the steps performed by (1). As in previous workflow the AI/ML MSs are generically indicated as AI/ML AI-based & Predictive Decision Service (E2E and not E2E) in a single box per domain, in order to ease the readability of the figures.

(1) Explicit Termination Request

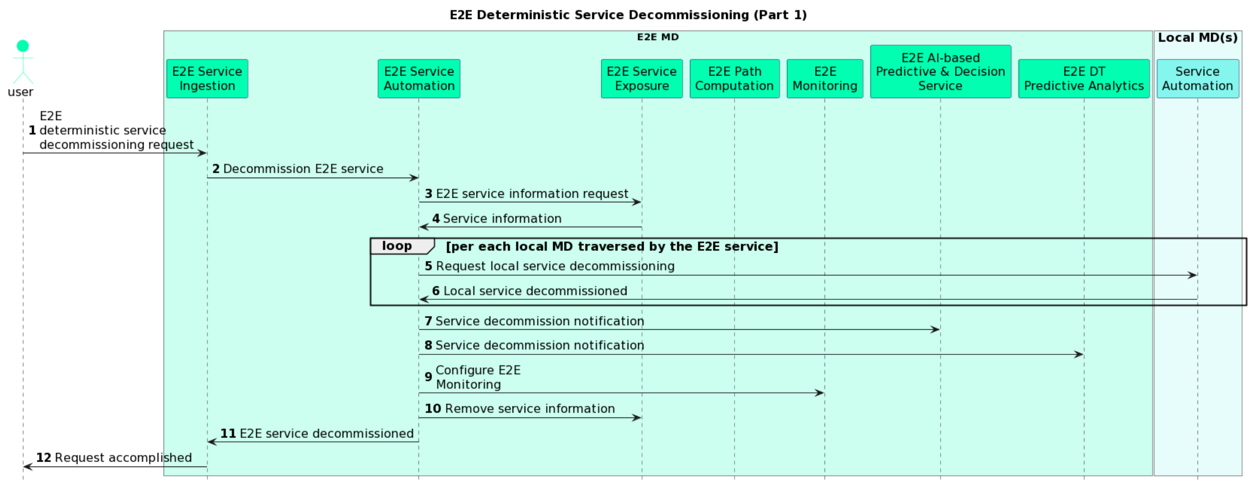

E2E Deterministic Service Decommissioning (Part 1)
E2E MD
Local MD(s)
user
E2E Service Ingestion
E2E Service Automation
E2E Service Exposure
E2E Path Computation
E2E Monitoring
E2E AI-based Predictive & Decision Service
E2E DT Predictive Analytics
Service Automation
1 E2E deterministic service decommissioning request
2 Decommission E2E service
3 E2E service information request
4 Service information
loop [per each local MD traversed by the E2E service]
5 Request local service decommissioning
6 Local service decommissioned
7 Service decommission notification
8 Service decommission notification
9 Configure E2E Monitoring
10 Remove service information
11 E2E service decommissioned
12 Request accomplished

Figure 8-4 E2E Deterministic Service decommissioning - Part 1, E2E Management Domain view

- **Step 1.** *User s*ends Service Decommissioning Request to *E2E Service Ingestion.*
- **Step 2.** *E2E Service Ingestion* forwards the request to *E2E Service Automation* where E2E service decommissioning is initiated.
- **Step 3.** *E2E Service Automation* retrieves additional service information from the *E2E Service Exposure* to perform the decommissioning.
- **Step 4.** *E2E Service Exposure* returns additional service information to the *E2E Service Automation.*
- **Step 5.** *E2E Service Automation* informs the *Service Automation MS* of the specific technology-domain about the service decommissioning and starts an iteration in the local MD.
- **Step 6.** *E2E Service Automation* is informed about the local service decommissioning and turns it into an E2E service decommissioning.

- **Step 7.** *E2E Service Automation* forwards the E2E Service Decommissioning notification to the *E2E Learning Orchestrator.*
- **Step 8.** *E2E Service Automation* forwards the E2E Service Decommissioning notification to the *DT Predictive Analytics.*
- **Step 9.** *E2E Service Automation* configures E2E monitoring for stopping the monitoring of the parameters related to the decommissioned service.
- **Step 10.** *E2E Service Automation* removes the E2E service information from the *E2E Service Exposure MS.*
- **Step 11.** *E2E Service Automation* informs the *E2E Service Ingestion* about the E2E Service Decommissioning.
- **Step 12.** *E2E Service Ingestion* notifies the user or operator about the success of the E2E Service Decommissioning request.

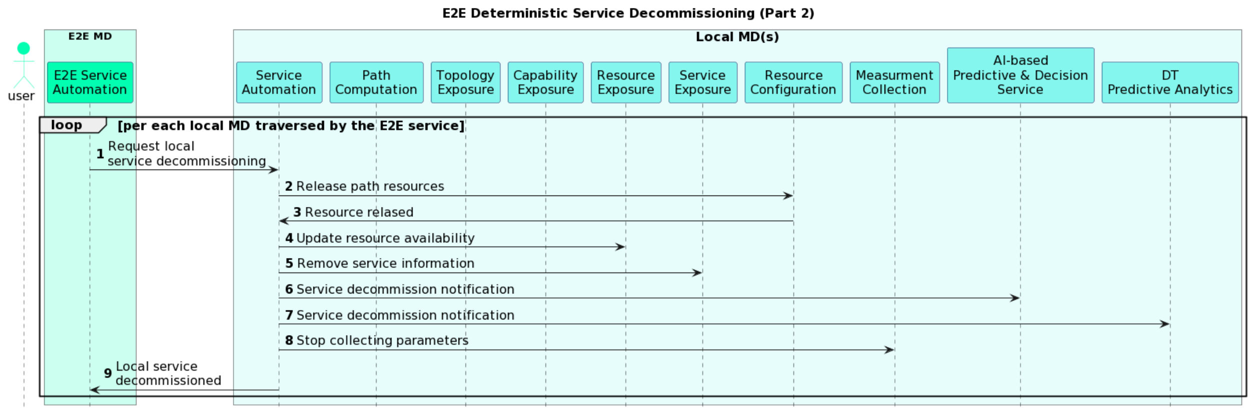

E2E Deterministic Service Decommissioning (Part 2)
E2E MD
Local MD(s)
user
E2E Service Automation
Service Automation
Path Computation
Topology Exposure
Capability Exposure
Resource Exposure
Service Exposure
Resource Configuration
Measurment Collection
AI-based Predictive & Decision Service
DT Predictive Analytics
loop [per each local MD traversed by the E2E service]
1 Request local service decommissioning
2 Release path resources
3 Resource relased
4 Update resource availability
5 Remove service information
6 Service decommission notification
7 Service decommission notification
8 Stop collecting parameters
9 Local service decommissioned

Figure 8-5 E2E Deterministic Service decommissioning - Part 2, Local Management Domain view

- **Step 1.** *Service Automation MS* receives the local service decommissioning request.
- **Step 2.** *Service Automation MS* requests to the *Resource Configuration MS* the release of the resources allocated to the specific service in the local domain and the specific path.
- **Step 3.** *Resource Configuration MS* releases the corresponding resources.
- **Step 4.** *Service Automation MS* informs the *Resource Exposure MS* about the update on the resource availability.
- **Step 5.** *Service Automation MS* removes service information from the *Service Exposure*.
- **Steps 6 and 7. Service Automation MS** informs about the service decommissioning to the *Learning Orchestrator* and the *DT Predictive Analytics MSs*.

- **Step 8.** *Service Automation MS* requests to stop collecting monitoring parameters to the *Measurement Collection MS*.
- **Step 9.** *Service Automation MS* informs the *E2E Service Automation MS* about the local service decommissioning.

(2) Expiration of the Service Lifetime

In this case, the E2E Service Automation automatically detects the service expiration and trigger its termination. The workflow coincides with (1) starting from step 3.

8.2 Appendix B – E2E Service lifecycle model

Figure 8-6 shows the set of administrative state in which a service can be. A more detailed description is available in D1.2 (PREDICT-6G/D1.2/5.4.3, 2023).

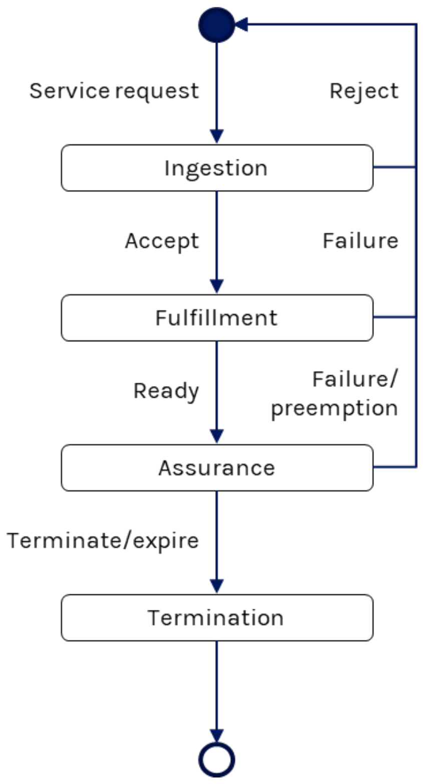


Figure 8-6. E2E service administrative states and state transitions

8.3 Appendix C – AICP information models

All the information reported in this Appendix are available in the project's GitLab[25].

8.3.1 Deterministic Services

8.3.1.1 E2E Deterministic Service request

```
module: E2E_deterministic_service_request_model
  +--rw e2e-service
     +--rw endpoints
     |  +--rw source
     |  |  +--rw id?   string
     |  +--rw destination
     |     +--rw id?   string
     +--rw qos-characteristics
     |  +--rw priority?                       uint8
     |  +--rw reliability?                    percent
     |  +--rw packet-loss?                    percent
     |  +--rw e2e-delay?                      milliseconds
     |  +--rw e2e-rtt?                        milliseconds
     |  +--rw jitter?                         milliseconds
     |  +--rw burst-arrival-time-window
     |  |  +--rw t1?   milliseconds
     |  |  +--rw t2?   milliseconds
     |  +--rw burst-completion-time-window
     |     +--rw t1?   milliseconds
     |     +--rw t2?   milliseconds
     +--rw traffic-characteristics
     |  +--rw direction?              enumeration
     |  +--rw periodicity?            boolean
     |  +--rw period?                 milliseconds
     |  +--rw burst-size?             uint32
     |  +--rw maximum-flow-bitrate?   thoughput
     +--rw tft
     +--rw service-lifetime
        +--rw service-duration
        |  +--rw t1?   milliseconds
        |  +--rw t2?   milliseconds
        +--rw recurrent-service-interval
           +--rw t1?                    milliseconds
           +--rw t2?                    milliseconds
           +--rw recurrence-interval?   milliseconds
```

8.3.1.2 E2E Deterministic Service data model

```
module: E2E_deterministic_service_model
```

[25] https://gitlab.netcom.it.uc3m.es/predict-6g/aicp/information-models

```
  +--rw e2e-service
     +--rw id?                         string
     +--rw service-status?             status-type
     +--rw endpoints
     |  +--rw source
     |  |  +--rw id?   string
     |  +--rw destination
     |     +--rw id?   string
     +--rw e2e-path-selection
     |  +--rw id?           string
     |  +--rw path-status?   string
     |  +--rw domain-list* [index]
     |     +--rw index          uint32
     |     +--rw path-domains
     |        +--rw name?    string
     |        +--rw class?   domain-type
     +--rw qos-characteristics
     |  +--rw priority?                        uint8
     |  +--rw reliability?                     percent
     |  +--rw packet-loss?                     percent
     |  +--rw delay?                           milliseconds
     |  +--rw rtt?                             milliseconds
     |  +--rw jitter?                          milliseconds
     |  +--rw burst-arrival-time-window
     |  |  +--rw t1?   milliseconds
     |  |  +--rw t2?   milliseconds
     |  +--rw burst-completion-time-window
     |     +--rw t1?   milliseconds
     |     +--rw t2?   milliseconds
     +--rw traffic-characteristics
     |  +--rw direction?               enumeration
     |  +--rw periodicity?             boolean
     |  +--rw period?                  milliseconds
     |  +--rw burst-size?              uint32
     |  +--rw maximum-flow-bitrate?   thoughput
     +--rw tft
     +--rw service-lifetime
     |  +--rw service-duration
     |  |  +--rw t1?   milliseconds
     |  |  +--rw t2?   milliseconds
     |  +--rw recurrent-service-interval
     |     +--rw t1?                    milliseconds
     |     +--rw t2?                    milliseconds
     |     +--rw recurrence-interval?   milliseconds
     +--rw md-services
        +--rw local-service* [index]
           +--rw index      uint32
           +--rw service
              +--rw id?               string
              +--rw service-status?   status-type
              +--rw target-domain
                 +--rw name?    string
```

```
                  +--rw class?   domain-type
```

8.3.1.3 Local Deterministic Service data model

```
module: local_service_model
  +--rw td-service
     +--rw id?                        string
     +--rw e2e-service-id?            string
     +--rw service-status?            status-type
     +--rw td-nodes
     |  +--rw endpoint
     |  |  +--rw id?               string
     |  |  +--rw node-resources
     |  |     +--rw resource-list* [index]
     |  |        +--rw index    uint32
     |  |        +--rw CPU?     cpu
     |  |        +--rw RAM?     ram
     |  |        +--rw disk?    storage
     |  +--rw gateway
     |     +--rw gateway-list* [index]
     |        +--rw index    uint32
     |        +--rw node
     |           +--rw id?               string
     |           +--rw node-resources
     |              +--rw resource-list* [index]
     |                 +--rw index    uint32
     |                 +--rw CPU?     cpu
     |                 +--rw RAM?     ram
     |                 +--rw disk?    storage
     +--rw td-path-selection
     |  +--rw path-id?          string
     |  +--rw path-status?      status-type
     |  +--rw involved-nodes
     |     +--rw node-list* [index]
     |        +--rw index    uint32
     |        +--rw node
     |           +--rw id?               string
     |           +--rw node-resources
     |              +--rw resource-list* [index]
     |                 +--rw index    uint32
     |                 +--rw CPU?     cpu
     |                 +--rw RAM?     ram
     |                 +--rw disk?    storage
     +--rw qos-characteristics
     |  +--rw priority?                       uint8
     |  +--rw td-reliability?                 percent
     |  +--rw td-packet-loss?                 percent
     |  +--rw td-delay?                       milliseconds
     |  +--rw td-rtt?                         milliseconds
     |  +--rw td-jitter?                      milliseconds
     |  +--rw burst-arrival-time-window
     |  |  +--rw t1?   milliseconds
     |  |  +--rw t2?   milliseconds
     |  +--rw burst-completion-time-window
     |     +--rw t1?   milliseconds
     |     +--rw t2?   milliseconds
     +--rw traffic-characteristics
     |  +--rw direction?             enumeration
     |  +--rw periodicity?           boolean
     |  +--rw period?                milliseconds
     |  +--rw burst-size?            uint32
     |  +--rw maximum-flow-bitrate?   thoughput
     +--rw tft
     +--rw service-lifetime
        +--rw service-duration
        |  +--rw t1?   milliseconds
        |  +--rw t2?   milliseconds
        +--rw recurrent-service-interval
           +--rw t1?                    milliseconds
           +--rw t2?                    milliseconds
           +--rw recurrence-interval?   milliseconds
```

8.3.2 Topology Exposure

8.3.2.1 E2E Topology Read request

```
{
         "topology:" {
                  "nodes": [
                           {
                                    "id": "123e4567",
                                    "type": "3GPP"
                           }
                  ],
                  "links": [
                           {
                                    "id": "373938c6",
                                    "source-termination-point": "5790f992",
                                    "destination-termination-point": "7aba13c2",
                                    "minimum-latency": "10",
                                    "maximum-latency": "100",
                                    "available-capacity": "150",
                                    "reliability": "99.99"
                           }
                  ]
}
```