



Human-centric algebraic machine learning

Overall Architecture Definition - Revised M18

Deliverable D1.2

**WP1 - Project Management,
Architecture and Technical Coordination**



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952091.

Consortium



ALGEBRAIC AI



Chantalimaud
Foundation



EPROSIMA
The Middleware Experts



German
Research Center
for Artificial
Intelligence



TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

uc3m



Project

Title: ALMA: Human Centric Algebraic Machine Learning
 Acronym: ALMA
 Coordinator: Proyectos y Sistemas de Mantenimiento SL (eProsima)
 Grant agreement: 952091
 Call: H2020-EIC-FETPROACT-2019
 Program: Horizon 2020
 Start: 01 September 2020
 Duration: 48 months
 Website: <https://alma-ai.eu>
 E-mail: alma@eprosima.com
 Consortium: **Proyectos y Sistemas de Mantenimiento SL (eProsima) (EPROS)**, Spain
Champalimaud Foundation (CF), Portugal
ALGEBRAIC AI (ALG), Spain
German Research Center for Artificial Intelligence (DFKI), Germany
Inria (Inria), France
TU Kaiserslautern (TUK), Germany
FIWARE Foundation e.V. (FF), Germany
Technical Research Centre of Finland (VTT), Finland
Carlos III University (UC3M), Spain

Deliverable

Number: **D1.2**
 Title: **Overall Architecture Definition - Revised M18**
 Month: 18
 Work Package: WP1 - Project Management, Architecture and Technical Coordination
 Work Package leader: Proyectos y Sistemas de Mantenimiento SL (eProsima)
 Deliverable leader: German Research Center for Artificial Intelligence (DFKI)
 Deliverable type: Report (R)
 Dissemination level: Public (PU)
 Date of submission: 2022-02-28
 Version: v2.1
 Status: Finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	2021-05-19	EPROS	Javier Paris	D1.1 - First draft
v1.1	2021-05-20	EPROS	Raúl Sánchez-Mateos Lizano	D1.1 - Second draft
v1.2	2021-05-24	ALG	Nabil Abderrahaman	D1.1 - Third draft
v1.3	2021-05-25	EPROS	Raúl Sánchez-Mateos Lizano	D1.1 - First Review (RV1)
v1.4	2021-05-27	CF	Gonzalo G. de Polavieja	D1.1 - Second Review (RV2)
v2.0	2022-02-15	EPROS	Juan López Fernández	D1.2 - First draft
v2.1	2022-02-28	EPROS	Raúl Sánchez-Mateos Lizano	D1.2 - Revision (RV1)

Executive summary

Algebraic Machine Learning (AML) is a novel machine learning technique based on algebraic representations of data. Unlike statistical learning, AML algorithms are robust regarding the statistical properties of the data and are parameter-free. AML systems can retain what they learn, including sociocultural ethical practices for human–artificial intelligence systems. The aim of the EU-funded ALMA project is to study the mathematical properties of AML and to leverage them to develop a new generation of interactive human-centric machine learning systems. These systems are intended to minimize bias in order to facilitate trust while better enabling distributed collaborative learning.

This report corresponds to *Deliverable D1.2 - Overall Architecture Definition - Revised M18* of the ALMA project. This deliverable covers the definition of the overall architecture of the AML environment i.e. it covers both the definition of the specification of the AML nodes required for the complete user execution of the AML algorithm, and the specification and design of the communications framework used by these nodes to work collaboratively.

First, the AML is introduced to describe the context and underlying assumptions of the deliverable. Then, the description of the AML Description Language (AML-DL) and the design of the AML Integrating Platform (AML-IP) are presented. Both sections are completed with the *Deliverable D3.2 - AML-DL Specification - Revised M18* and *Deliverable D6.2 - AML-IP implementation - Revised M18*. Moreover, the current state of the AML environment is shown, i.e. the current state of the AML-DL and the integration of the AML nodes in the AML-IP.

Finally, the innovation management plan is included. It shows the procedures to be complied with by the members of the Consortium in the publication of the knowledge generated within the ALMA project.

Revision changes

Revision of deliverable *D1.1 - Overall Architecture Definition - Initial* for *Work Package 1: Project Management, Architecture and Technical Coordination*.

- Correction of typos.
- In subsection 2.1, *parallel learning* is renamed to *collaborative learning* in order to remain consistent with *Deliverable D6.2 - AML-IP implementation - Revised M18*.
- Section 5 has been extended with a brief explanation of the design made for the AML-IP in a *workload distribution scenario* (subsection 5.4).
- Updated AML-IP communication and nodes layer model present in subsection 4.2, now including the newly introduced AML-IP Algebraic Main Node (AML-IP-Main) and AML-IP Algebraic Computing Node (AML-IP-Computing).
- *eProxima Integration Service* has been replaced by the *DDS Router* as the chosen tool for bridging the connection between different Local Area Networks (LANs).

Contents

Executive Summary	3
Revision Changes	4
Contents	5
Acronyms	6
1 Introduction	7
1.1 Purpose of document	7
1.2 Partners involved	7
2 Algebraic Machine Learning	8
2.1 AML environment	8
2.1.1 Communication between AML nodes	8
3 AML Description Language	10
4 AML Integrating Platform	10
4.1 Dependencies	10
4.2 AML-IP Design	11
4.3 AML-IP functional operation	13
4.3.1 Phases of communication	13
4.3.2 AML-IP operational area networks	13
5 Current State	14
5.1 AML Integrating Platform Prototype	15
5.1.1 Prototype Nodes	15
5.1.2 Data types	16
5.1.3 Experiments and Results	17
5.2 AML nodes	17
5.3 Integration of AML nodes as AML-IP-Alg components	17
5.4 AML-IP workload distribution scenario	18
6 Innovation Management	19
7 Conclusions	20
8 Future Work	21
References	22

Acronyms

AI	Artificial Intelligence.
AML	Algebraic Machine Learning.
AML-CC	AML Consistency Checker.
AML-DL	AML Description Language.
AML-Engine	AML Engine.
AML-IP	AML Integrating Platform.
AML-IP-Agent	AML-IP Agent Node.
AML-IP-Alg	AML-IP Algebraic Node.
AML-IP-Computing	AML-IP Algebraic Computing Node.
AML-IP-D	AML-IP Discovery Node.
AML-IP-Engine	AML-IP Algebraic Engine Node.
AML-IP-Main	AML-IP Algebraic Main Node.
DDS	Data Distribution Service.
GA	Grant Agreement.
IGD	Internal Gateway Device.
LAN	Local Area Network.
ML	Machine Learning.
MSCP	MultiService Communication Protocol.
NAT	Network Address Translation.
P2P	Peer to Peer.
PO	Project Officer.
PoC	Proof of Concept.
Proto-Alg-DL	AML-IP Prototype Algebraic Description Language Node.
Proto-Alg-Engine	AML-IP Prototype Algebraic Engine Node.
Proto-DiscoveryServer	AML-IP Prototype Discovery Server Node.
TCP	Transmission Control Protocol.
UPnP	Universal Plug and Play.
WAN	Wide Area Network.
WWW	World Wide Web.

1 Introduction

The ALMA research and innovation project aims to develop a novel approach to the Artificial Intelligence (AI) paradigm by implementing and further exploring Algebraic Machine Learning (AML), a new Machine Learning (ML) technique with a strong foundation in Abstract Algebra and Model Theory. The goal of the EU-funded ALMA project is to leverage the properties of AML to develop a new generation of interactive, human-centered machine learning systems.

1.1 Purpose of document

This document aims to describe the implementation and integration of a platform for the massively distributed, massively parallel execution of AML algorithms. In particular, this document focuses on the Overall Architecture of the project. Here, a summary of the components, how they are connected for sharing data, and the current state of the project are presented. The main elements that will form the architecture for executing AML are the **AML Engine (AML-Engine)** itself, the **AML Description Language (AML-DL)**, the **AML Consistency Checker (AML-CC)**, and the **AML Integrating Platform (AML-IP)**.

The AML Engine (AML-Engine) carries out the learning process, creating models from data and formal knowledge. It executes the AML algorithm, increasingly building the model as it learns and generalises knowledge from examples and pre-established rules. Several AML-Engine can work on the same problem in parallel, independently building partial models and periodically sharing knowledge. The final model can then be built by aggregating and combining all partial models.

The main goal of AML-DL is to provide the users with a language to describe ML problems for AML, using algebraic semantic embeddings, in a concise and expressive manner. An embedding is a set of positive and negative relations between pairs of elements, that constitute the input data for the AML-Engine. AML-DL and an overview of semantic embeddings can be found in *Deliverable D3.2 - AML-DL Specification - Revised M18*.

The AML Consistency Checker (AML-CC) verifies that a set of relations describing an AML problem are all consistent and there are no two rules contradicting each other. This is a necessary step before transferring data to an engine. In essence, a training batch cannot contain two relations of the form $A \text{ equal } B$ and $A \text{ not equal to } B$. However, note that this only affects examples within the same batch, the dataset can contain any number of errors, contradictions and mislabels, as long as they are computed in different training batches. The AML-CC is further explained in *Deliverable D3.8 - AML-DL Consistency Checker - Initial*.

The AML-IP is the platform that manages the whole communication process between the different nodes that form the AML environment. This platform creates an Internet wide network and handles communication between nodes in an efficient and distributed way, allowing to create a parallel, scalable AML architecture. It also offers a user friendly framework to easily create AML internal or external networks. This element is explained in section 4, and more detailed about its design and its actual state could be found in this project's deliverable *Deliverable D6.2 - AML-IP implementation - Revised M18*.

1.2 Partners involved

Deliverable Leader:

- German Research Center for Artificial Intelligence (DFKI), Germany

Partners:

- Algebraic AI (ALG), Spain

- Proyectos y Sistemas de Mantenimiento SL (eProsima) (EPROS), Spain

2 Algebraic Machine Learning

AML is a novel approach to ML grounded on Abstract Algebra and Model Theory, with no optimization procedure and with a lesser dependency on statistics [1, 2, 3]. One of its most appealing aspects, from an industrial point of view, is its capability for distributed computation, where the algorithm can theoretically scale up with no upper limit [4]. Thereby, the training dataset can be split into subsets, with the AML algorithm working on different subsets of the same problem in parallel and asynchronously across several machines. The final model can then be built by aggregating the individual contributions [4].

Note that parallel, multi-processor computing architectures are widely used in other ML methods, by running on a single machine with access to multiple processing units. This, however, does not leverage the whole potential for parallelism of AML. With a completely decentralised architecture, the AML algorithm is not limited to a single computer and can instead build models independently in several machines that are loosely coupled. These machines only need to submit their resulting model once their training phase is completed, additionally they have the option to communicate knowledge asynchronously during training to boost learning [4].

The AML-IP aims to exploit this property, creating an environment for independent building blocks of the AML algorithm to communicate and share data and results.

2.1 AML environment

The AML environment is comprised of nodes of different kinds that communicate with each other. These nodes compute different types of data that are published and shared with other nodes in the AML ecosystem that request them.

One of the revolutionary aspects of AML is its great parallelization capability, allowing different kinds of nodes to share their data to distribute the workload, and share their knowledge to boost each other's learning [4]. This implies that parallelization can be seen from two different angles:

- **Distributed workload:** The creation, processing and consumption of data is completely delocalized. Some nodes can create AML data that can later be processed by remote nodes in parallel executions.
- **Collaborative learning:** Different nodes can work on the same problem independently, using for instance different datasets. Sporadically, they can share the knowledge learnt with the other nodes, boosting their learning process and improving the accuracy of the model. Note that this also opens the door to the use of private datasets, as knowledge is shared but not the data itself.

2.1.1 Communication between AML nodes

The architecture on which this project focuses is based on the communication of different **AML nodes** through the Internet, freely distributing data and knowledge among them. The main points to be taken into account in order to properly understand AML-IP are:

- **Distributed architecture:** There is no main access point to the network, nor is there a centralised server to provide communication. This means Peer to Peer (P2P) communication is required between different AML nodes.
- **Data Distribution Service (DDS):** DDS will be the main communication protocol between nodes of the network.

- **World Wide Web (WWW):** This architecture is deployed over the Internet, allowing anyone with internet access to be part of the network.

Figure 1 presents a simple example of an AML execution. There are several nodes along the network that work under a DDS topic. Each node can publish under these DDS topics and/or subscribe to them, communicating with all other nodes within the AML network working on these same topics. Thus a P2P communication is achieved in which communication is not centralised.

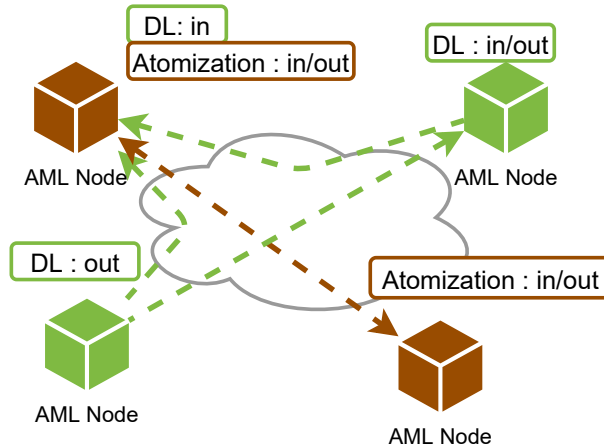


Figure 1: AML Nodes communicating with each other.

In the scenario shown in Figure 1, two types of AML nodes are presented. The green nodes communicate through the DL topic. This topic stands for the AML-DL, the language used by all AML nodes to communicate with each other. Moreover, the brown nodes represent AML computational engines. They produce a different type of data known as atomization. This atomization data can be either the input or the output of the node. Therefore these nodes can both publish this data type under the Atomization DDS topic and/or subscribe to it.

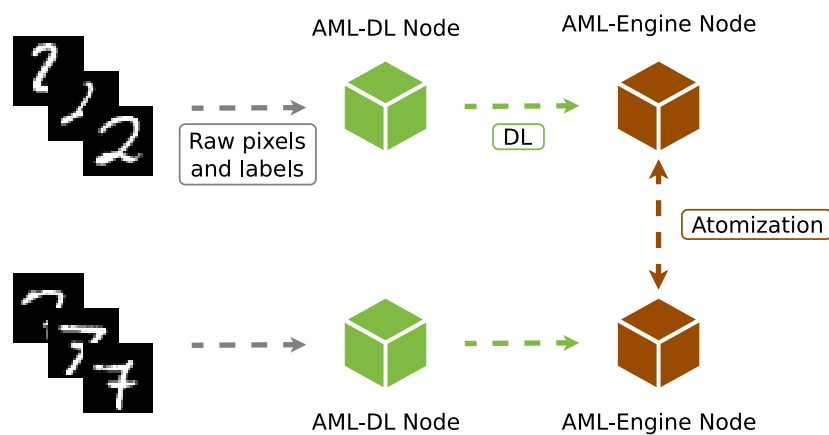


Figure 2: AML Nodes communicating with each other (MNIST dataset).

Figure 2 describes a network that learns how to identify handwritten numbers, for instance those in

MNIST¹. Raw data is first passed to one or many AML-DL nodes, that interpret and publish it to the network using the DL topic. This data is then consumed by AML-Engine nodes that begin the learning process and also publish to and consume from the Atomization topic. To illustrate how data can be distributed, this example assumes that one single AML-Engine node only consumes samples corresponding all of them to the same label in the training data set, that is they only learn to identify one single number. For instance, all samples labelled as "2" will be consumed by a single AML-Engine node and only by it. At the end of the training process, each AML-Engine node has a model able to identify one single number. These models can then be combined into one to be able to identify all 10 digits.

3 AML Description Language

The goals of AML-DL are to give the user the capability to describe complex algebraic semantic embeddings in a simple and concise way, but also to establish a methodology to build embeddings for real life problems. A small set of commands allows for the expression of large algebraic embeddings with brief AML-DL scripts. By using nested iterator commands a single AMLDL statement can translate into a large set of relations. This has already proven extremely useful in practice.

Although AMLDL may never be able to fully remove the need for a user with experience, there are primitives and methodologies that are integrated into AMLDL to make the embedding process more intuitive and streamlined. The goal of the language is to be flexible and adaptable, so the process is simple for the user. The embedding of a problem often requires its dissection into smaller parts. Some of which are common to many problems and these common parts are motifs for which specific AMLDL syntax could be provided, and so improving and expanding the language over time.

This language and the mathematical theory on which it is based is fully described in *Deliverable D3.2 - AML-DL Specification - Revised M18*.

4 AML Integrating Platform

The AML-IP is a communications framework in charge of data exchange between AML nodes through the Internet, which focused on the easily deployment of AML nodes and the communication between them. Thus, AML-IP is not just an user oriented application, but an intermediate library that makes the transport protocols transparent to the user. Therefore, it is thought to create a platform that communicates each node with those that are in the same AML network without requiring the user to know or understand the transport protocols behind each communication.

4.1 Dependencies

This section describes the AML-IP technical dependencies. All of them focus on the communication between nodes, the protocols used to support such communication and the tools used to handle the different types of data to be transmitted.

- ***eProxima Fast DDS***. Given the distributed nature of the AML ecosystem, the use of DDS is proposed to enable real-time communication between AML nodes. *eProxima Fast DDS* is a C++ implementation of the DDS specification and provides both an Application Programming Interface (API) and a communication protocol to deploy a DDS environment.

¹The MNIST Database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>

Moreover, the dynamic addition of new entities to the network is also covered by the the DDS protocol, allowing the dynamic discovery of new instantiated DDS entities. The AML-IP framework will exploit the Discovery Server discovery mechanism described in [5].

Fast DDS also implements the TCP transport protocol for data transmission between DDS entities in Wide Area Networks (WANs). The purpose of the AML-IP is to implement a communications framework that uses the Transmission Control Protocol (TCP) communications protocol in a transparent manner for the user.

- ***eProxima DDS Router***². This is an end-user software application that enables the connection of distributed DDS networks. That is, DDS entities such as publishers and subscribers deployed in one geographic location and using a dedicated local network will be able to communicate with other DDS entities deployed in different geographic areas on their own dedicated local networks as if they were all on the same network.
- **miniUPnP**. This is a C library that supports the Universal Plug and Play (UPnP) Internal Gateway Device (IGD) specifications. Due to the distributed nature of the AML environment, AML nodes require a mechanism to be accessed from outside their LAN. This is made possible by port forwarding of the user's private IGD. The aim is to provide AML-IP nodes with the capability to perform this action automatically and transparently to the user.

A more detailed explanation of these technologies can be found in *Section 2.5 Enabling technologies of Deliverable D6.2 - AML-IP implementation - Revised M18*.

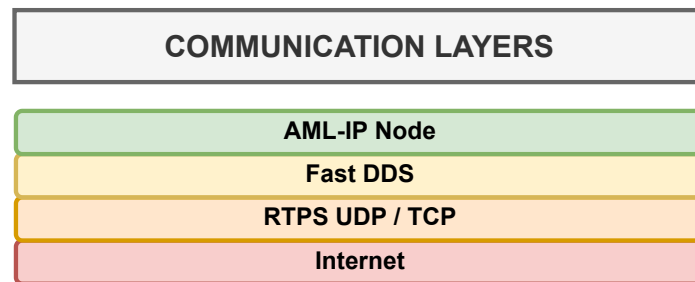
4.2 AML-IP Design

The chosen design is focused on the division of the different AML-IP components that constitute the network. In order to divide the workload, to distribute the communication process and to create an scalable and widely usable architecture, the components of the AML-IP are divided in three categories, i.e. the AML-IP architecture consists of three types of nodes:

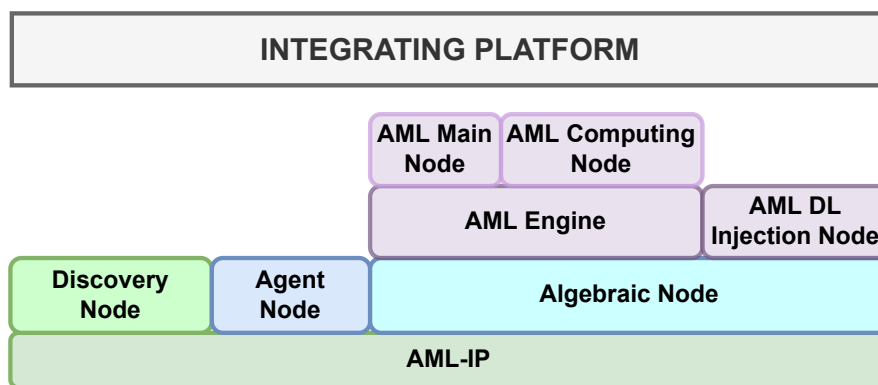
- **AML-IP Algebraic Node (AML-IP-Alg)**: These nodes represent each of the instances of the AML.
- **AML-IP Agent Node (AML-IP-Agent)**: This is the main transport component that manages the dynamic discovery of AML-IP Algebraic Nodes (AML-IP-Algs) deployed within a LAN and the discovery of other AML clusters over WAN.
- **AML-IP Discovery Node (AML-IP-D)**: These components will be disseminated along the network and made accessible through the Internet. They manage the WAN discovery process along the whole AML environment, handling the discovery of independent AML clusters and efficiently connecting the AML-IP-Alg that must connect to each other.

The general architecture of the AML-IP is displayed in Figure 3. It shows how the AML-IP would integrate all transport and network elements (3a), as well as all types of AML-IP nodes and the hierarchy among them (3b).

²<https://github.com/eProxima/DDS-Router>



(a) AML-IP communication layer model.



(b) AML-IP nodes layer model.

Figure 3: AML-IP Architecture

Figure 4 shows a general example of an AML network and how the AML-IP will establish a communication method for the distributed nodes. The AML-IP Discovery Nodes (AML-IP-Ds) are network nodes that are distributed along the Internet and work as a *well known address* in order to manage the WAN discovery. Other kind of nodes are the AML-IP Agent Nodes (AML-IP-Agents), network nodes deployed by each user. They manage the connection between a LAN cluster and the rest of the network. The AML cluster is just a group of nodes inside the same LAN that can communicate with each other within the LAN to which they belong in a faster and more efficient way than through WAN.

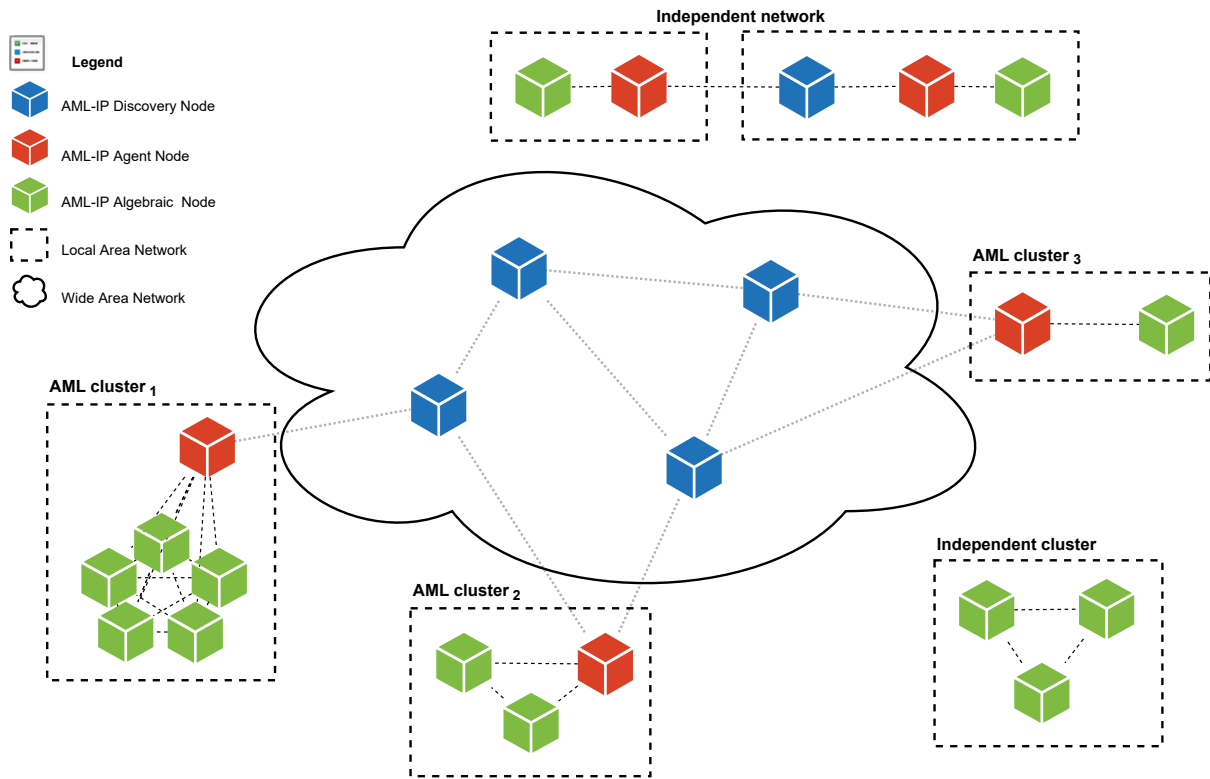


Figure 4: AML-IP example in LAN and WAN networks.

4.3 AML-IP functional operation

This section explains and exemplifies the communication process of all nodes within the the AML-IP. The differences between the various stages of inter-node communication are also explained.

4.3.1 Phases of communication

The communication process in DDS is divided in two phases: the discovery phase and the data transfer phase:

- **Discovery phase:** The discovery phase establishes the mechanism by which new DDS entities are detected by other entities already operating in the network. New entities communicate their arrival in the network via messages to a multicast group, and continue sending these messages in order to discover new entities or inform known entities of their current status. These messages are required to maintain the network state in a dynamic and distributed way.
- **AML data transfer phase:** This phase corresponds to the transfer of data required by the AML-IP-Alg to correctly execute their function.

4.3.2 AML-IP operational area networks

The networks where the AML-IP could operate are the Local Area Network (LAN), whether all the hosts are under the same IGD and share a private network; or the Wide Area Network (WAN), whether nodes are distributed along a wide network and usually behind different Network Address Translations (NATs).

In the figure 5 it is represented an example of communication between different kinds of nodes deployed in LAN and WAN. The discovery and the data transfer phases depends on the network type. The nodes that are under the same LAN discover each other by *DDS Simple Discovery*, that is, by multicast, and communicate to each other independently of the rest of the internal or external network. In order to know the existence of external clusters whit nodes that require to interchange data, it must be deployed an AML-IP-D network that interconnects different LANs by distributing the discovery information. To manage the WAN discovery and data communication, an AML-IP-Agent must be deployed in every LAN and it centralizes the whole process to WAN communication. Two different AML-IP-Agent communicate to each other in P2P. They do not relay on the AML-IP-Ds that has connected them except to know the actual state or possible updates on the other network.

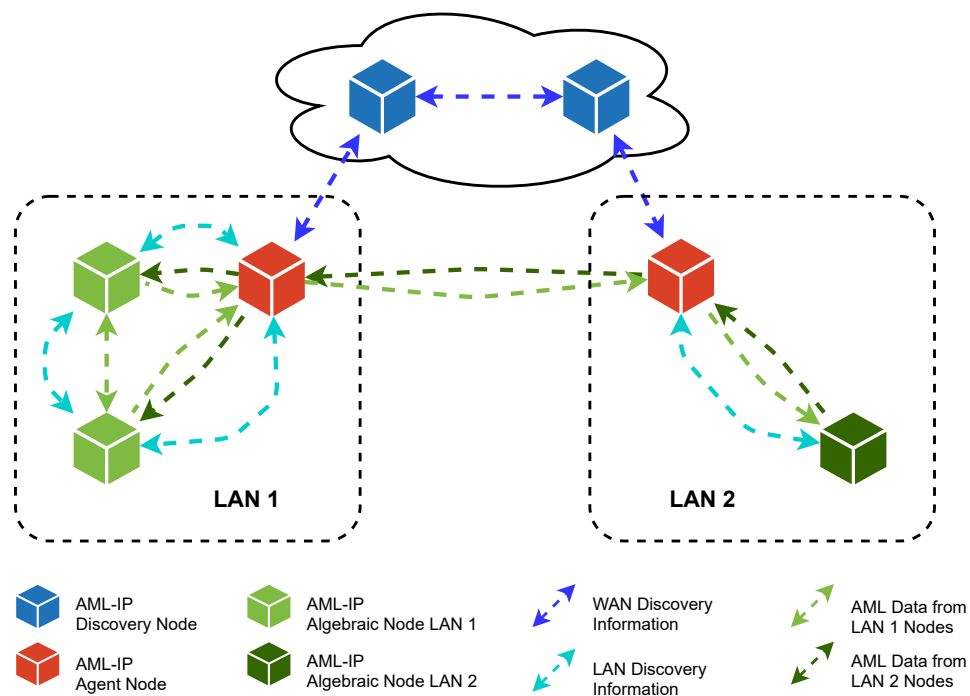


Figure 5: AML-IP minimum example to describe LAN and WAN communication.

A more detailed explanation of the functionality and design of each of the AML-IP components, as well as the detailed steps necessary to establish communication between them can be found in *Deliverable D6.2 - AML-IP implementation - Revised M18*.

5 Current State

This section presents the current status of the overall ALMA project architecture. It starts with a brief explanation of the prototype of the AML-IP communications framework; then the current status of the actual AML nodes that compute the AML algorithm is presented; following, the integration of AML nodes, such as the AML-Engine and AML-DL, as elements of the AML-IP is explained; it concludes with a summary of the design made for the AML-IP with the aim of demonstrating the parallelization capabilities of AML in workload distribution scenarios.

5.1 AML Integrating Platform Prototype

In order to demonstrate the validity and usability of the design of the AML-IP proposed in this document, an AML-IP Prototype has been developed. It implements the key aspects of the AML-IP that make the prototype a minimum functional version of the final communications framework. This AML-IP prototype is publicly available and is available in the GitHub repository **aml-ip-prototype**³ under the Open Source Apache 2.0 license⁴.

This prototype is not aimed to be a robust final tool, but rather a quick-to-implement tool to serve as a Proof of Concept (PoC) for future development of the AML-IP communications framework. The idea of the prototype is to show a minimal example of DDS communication through WAN with various AML-IP-Alg discovered by one or multiple AML-IP-D.

5.1.1 Prototype Nodes

There are three AML-IP nodes in the prototype to be differentiated:

- **AML-IP Prototype Algebraic Description Language Node (Proto-Alg-DL):** These nodes will represent every node in the network that publish under a *Description Language* topic. The output of these nodes will be *Proto-Alg-DL relations* that the AML algorithm can understand. This node is the AML-IP-Alg representation of the AML-IP components described in *D6.1 - AML-IP implementation*.
- **AML-IP Prototype Algebraic Engine Node (Proto-Alg-Engine):** This node represents the AML node in charge of processing and executing the AML algorithm. It is the node with the highest workload. The input data-stream of this nodes will be the output data of the Proto-Alg-DL nodes. Its output data is the *atomization*, the knowledge built from the data given. This node is the AML-IP-Alg representation of the AML-IP components described in *D6.1 - AML-IP implementation*.
- **AML-IP Prototype Discovery Server Node (Proto-DiscoveryServer):** This nodes will execute the discovery process that will join the different nodes to those who are related, ergo, to those that must share information that is relevant. It represents an AML-IP-D.

³https://github.com/eProsima/aml-ip-prototype/tree/ALMA_AMLIP_D6.1

⁴https://github.com/eProsima/aml-ip-prototype/blob/ALMA_AMLIP_D6.1/LICENSE

DomainParticipant	DataWriters	DataReaders	Topic	
			Name	Data Type
Proto-Alg-DL	DL		<code>_aml_ip_topic_dlooutput</code>	<i>AML_IP_DLOutput</i>
Proto-Alg-Engine	Atom		<code>_aml_ip_topic_atomization</code>	<i>AML_IP_Atomization</i>
		Atom	<code>_aml_ip_topic_atomization</code>	<i>AML_IP_Atomization</i>
		DL	<code>_aml_op_topic_dlooutput</code>	<i>AML_IP_DLOutput</i>
Proto-DiscoveryServer				

Table 2: Overview table of user AML-IP node types. The first column shows the name of the node, which corresponds to the name designated for the DomainParticipant. The next two columns specify the DataWriter(s)/DataReader(s) of the DomainParticipant. Finally, the topic of each DataWriter/-DataReader is defined by its name and data type. Since the *AML_IP_DiscoveryServer* node is only involved in the entity discovery process, i.e. it does not apply to the data transfer phase, it does not contain any user DataReader/DataWriter.

5.1.2 Data types

The data will be transmitted using *Fast DDS* communications middleware which implements the DDS standard. In a real scenario, the data to be transmitted will depend on the type of the problem on which the AML nodes are working, using different DDS topics to differentiate the type and content of the data. However, for the case of the prototype it is assumed that all AML nodes work on the same ML problem. In the case of the prototype, the type of data ingested and generated by each type of node will be different. This results in two types of data:

- **DL** (*AML_IP_DLOutput*): Output data from AML-DL nodes. This data type contains a sequence of data of type *AML_IP_Relation*. The *AML_IP_Relation* represents the relation of a Term with a number of Constants. It is formed by two sequences of integers of different sizes, and a boolean which specifies the AML relation type.
- **Atomization** (*AML_IP_Atomization*): Output data from AML-Engine nodes. It represents the atomization learnt from an execution of the AML algorithm. This data type contains a sequence of data of type *AML_IP_Atom*. The *AML_IP_Atom* data type describes an Atom. It is formed by a sequence of integers of fix sized.

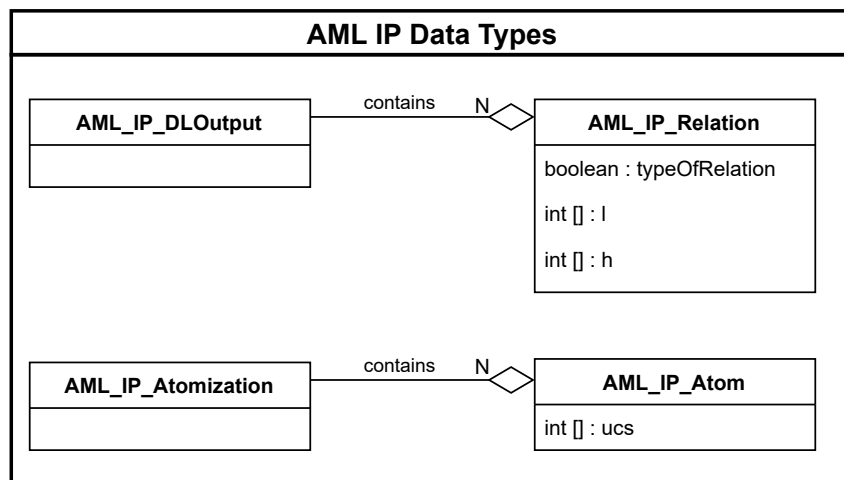


Figure 6: Data Models.

5.1.3 Experiments and Results

Several experiments have been built and run in order to test the LAN and WAN communication with various network architectures, configurations, and transport methods. The focus of this section is to explain the experiments execution and discuss the results. For more detailed information on these experiments please refer to *Deliverable D6.2 - AML-IP implementation - Revised M18*.

5.2 AML nodes

The AML-IP Prototype is currently tested with DDS entities that simulate the data generation, transmission and ingest behaviour of the AML-IP Algebraic Nodes. Although simulated for demonstrations purposes in this document, the actual implementation of the nodes is at a very mature stage, essentially only missing the DDS functionalities to be integrated in the AML-IP.

- **The AML-Engine node:** There is a fully functional AML algorithm implemented in both Python and C++.
- **The AML-DL node:** The first release of the specification is presented in *Deliverable D3.2 - AML-DL Specification - Revised M18*. An implementation of this specification exists, also in Python and C++.
- **The AML-CC node:** This node is in a completed state save for the DDS functionalities. A working demonstrator with examples is provided in *Deliverable D3.8 - AML-DL Consistency Checker - Initial*.

5.3 Integration of AML nodes as AML-IP-Alg components

Both the design of the AML-IP and its components follow a user-centered design. That is, it has been designed to facilitate the integration of the transport layer with existing AML nodes. Thus, the developer only needs to understand the basic concepts of the communications middleware to communicate the AML nodes. The aim of the AML-IP-Alg component is to make the integration of every AML component into the AML-IP as easy and fast as possible for the developer, while keeping the integration process independent of the type of AML node to be deployed. The AML-IP-Alg use *eProsima Fast DDS* as the communication middleware to communicate using DDS with all the nodes within its LAN, included a possible AML-IP-Agent that will connect it with nodes nodes deployed out of its LAN. Though, the only integration that needs to be done from the point of view of an AML developer is to specify a DDS topic under which an AML node will publish or subscribe. An example of how an AML-Engine node publishes random data that represents an *AML_IP_Atomization* can be found in the source code of the AML prototype publicly available on the GitHub repository `aml-ip-prototype`⁵ and described in *Deliverable D6.2 - AML-IP implementation - Revised M18*. In this same repository there is also an example of how the same AML-Engine node subscribes to and receives data from a specific topic^{6 7}.

The integration of the AML nodes as AML-IP-Alg nodes in the AML-IP communications framework is a future work to be done collaboratively between the AML-DL and AML-IP technical managers, i.e.

⁵https://github.com/eProsima/aml-ip-prototype/tree/ALMA_AMLIP_D6.1

⁶AML-Engine publication example:
<https://github.com/eProsima/aml-ip-prototype/blob/c6d6de24de7b3201082cb53efba49e01e733c09f/src/cpp/Engine/EngineParticipant.cpp#L240>

⁷AML-Engine subscription example:
<https://github.com/eProsima/aml-ip-prototype/blob/c6d6de24de7b3201082cb53efba49e01e733c09f/src/cpp/Engine/EngineParticipant.cpp#L342>

between **ALG** and **EPROS**. When one of the AML node types is fully defined, the communication between them using *Fast DDS* will be rapidly achieved. In order to fully define an AML node for its integration in the AML-IP, it needs to be defined the data type that this node produces/ingests, the name of the topic under its data will be published or the requested data will be received, and when this communication must be performed during the node operation.

The rest of the AML-IP components do not require any integration with any other nodes, as they only manage the discovery and communication between nodes in different WANs. Therefore, the rest of the AML-IP components, apart from the AML-IP-Alg nodes, are completely independent of the rest of the AML architecture.

5.4 AML-IP workload distribution scenario

The AML algorithm can be executed in a non-distributed way by running the whole computation on a single machine. For this purpose, parallel, multi-processor computing architectures can be exploited in a similar fashion to how other ML methods deal with parallelism. This, however, would not leverage the whole potential for parallelism of AML. With a completely decentralised architecture as the one described in this manuscript, the AML algorithm is not limited to a single computer and can instead run independently in several machines.

The AML-IP is a platform designed to cover a wide range of actions in an AML network; it allows for the collaboration between different networks, keeping the privacy and independence of each node; it makes possible to parallelize certain tasks between several local or remote nodes; it allows to merge several trained models in order to obtain a better one; and it also contemplates those use cases where multiple nodes may work jointly on the same task, by facilitating the communication and synchronization between nodes.

The so called *workload distribution scenario* refers to those scenarios in AML where one computational complex task could be divided in subtasks, and these subtasks could be processed in parallel by different computational points, to finally obtain a joint result. The workload distribution scenario directly applies to the training process performed in the AML-IP Algebraic Engine Node (AML-IP-Engine), the one in charge of training a model by adding new atomizations resulting from the processing of batches of data. This is a very expensive process that requires a high amount of memory and computational power. But it is also a parallelizable process, where some batches of data could be trained in parallel, and then the results could be merged to obtain a better model.

The main idea of this part of the AML-IP is to parallelize the computational effort in a balanced way between an arbitrary number of computing points. The work that is presented along this section refers to the process of creating new atomizations, generated by an AML-IP-Engine. It could be said that the scenario here exposed comprehends the detailed process of an AML-IP-Engine distributing its own workload among a number of computing points.

The nodes that take part in this sort of scenarios are the following:

- **AML-IP-Main:** This kind of node functions as a central point for a *training* process. It creates *Jobs*, that could be seen as batches of data to be processed by the training algorithm. These jobs are distributed to a number of computing nodes (as any batch could be trained in parallel thanks to the AML), and their corresponding results collected and merged back in the main node.
- **AML-IP-Computing:** This kind of node is in essence a computational point where a workload is released, so it can process a specific task on the data it has been given. This task corresponds to training over a batch of AML data and sending the pertinent result to the AML-IP-Main.

Figure 7 represents the previously described scenario. In this example, two AML-IP-Main work in the same problem, but with different datasets. Each of them has two different batches of training data to

process, but since an initial preprocessing phase is needed, the two batches are not required to be executed at the exact same time. With this schema it can be accomplished that, using three AML-IP-Computing, the workload of the two AML-IP-Main is distributed among them in an efficient way, so a task being resolved by a machine and other executions do not overlap. This is a specific use case, but any scenario with an arbitrary number of AML-IP-Main and AML-IP-Computing nodes is possible, and the workload would be *intelligently* divided between available and free nodes so the parallelization of complex tasks (as, in this case, the training phase) is maximized.

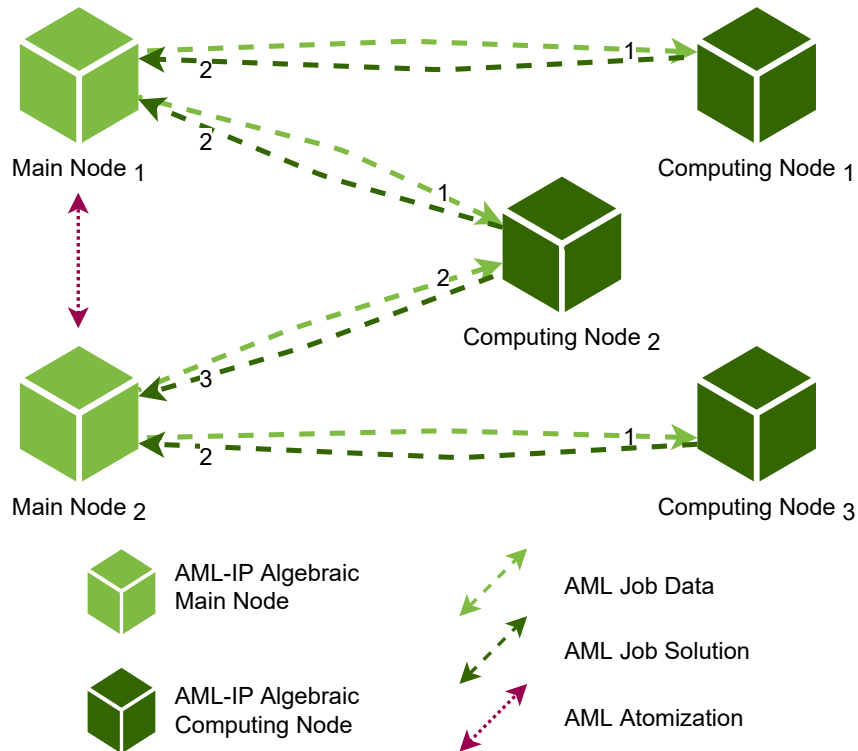


Figure 7: AML-IP workload distribution scenario.

Please refer to *Deliverable D6.2 - AML-IP implementation - Revised M18* for an in-depth description of the solution proposed for the *workload distribution scenario*, including the specification of a communications protocol (MultiService Communication Protocol (MSCP)) developed to handle the data exchange between these new kinds of nodes, and a prototype demonstrating the feasibility of the approach.

6 Innovation Management

According to *Section 3.2.2 Innovation Management* of the description of work included in the Grant Agreement (GA) 952091 of the ALMA project, "the partners in the Consortium undertake to provide a maximum level of transparency in terms of the available knowledge for the purposes of effective and efficient project execution".

The measures taken to ensure maximum awareness of the Project Officer (PO) of the available knowledge and internal communication within the ALMA Consortium are:

- *"Setting up and maintaining a project web page that will have a public section to communicate, for example, ongoing activities and project progress, a list of recent publications and events. As described in Section 3 Open access policy of ALMA of deldmp, all information on the public knowledge produced and the planned schedule of dissemination activities of the ALMA project will be published on the ALMA website⁸.*
- *"A private section to facilitate communication within the Consortium, e.g. by means of mailing lists and a document management section." This measure is described in Section 5.2 Making data openly accessible of Deliverable D8.10 - Data Management Plan - Initial.*
- *"Setting up and maintaining a project repository for publications, software, etc." As in the previous point, this measure is described in Section 5.2 Making data openly accessible, Section 5.4 Increase data re-use, and Section 6 Allocation of resources of Deliverable D8.10 - Data Management Plan - Initial.*

With regard to the dissemination and exploitation of the knowledge generated in the ALMA project, the ALMA Consortium will comply with the directives outlined in *Section 3 Open access policy of ALMA of Deliverable D8.10 - Data Management Plan - Initial*. As stated in this deliverable:

all deliverables already defined and the data generated will be publicly available, except for the ethical reports, whose distribution will be private and internal to ALMA consortium members. The research data will be uploaded to a public domain repository, and/or published in scientific journals, thus making it openly accessible. (...)

However, for each of the datasets and knowledge generated, the ALMA consortium will analyze the potential conflicts of openly publishing the results versus the commercialization and application of IPR protection, thus deciding what information will be public and when it will be public. (...)

7 Conclusions

This deliverable has described the general architecture of the AML environment, both the AML nodes that comprise it, as well as the general design of the AML-IP and the elements that constitute it. This is intended to be a living document so that it will be updated with the results and problems encountered in the implementation of each of the parts described in it.

Furthermore, AML-IP has been presented as a communications framework that exploits the collaborative and distributed learning capabilities of AML. It will create a network of AML clusters working collaboratively in different geographical locations over the Internet with minimal user configuration. This raises challenges such as the dynamic discovery of AML nodes working on the same problem and the communication between them over WANs. Thus, users working on the same AML problem will be able to share their learning results without requiring any knowledge of communication protocols.

This communication framework aims to build the AML environment with a focus on network scalability and communication security, taking into account constraints such as data centralisation at intermediate nodes and the large amount of data that needs to be transmitted.

Finally, the innovation management plan is presented. This defines the guidelines to be followed in the publication of the ALMA project outcomes, as well as the measures taken to achieve the highest level of transparency of the work performed by the partners towards the PO.

⁸<https://alma-ai.eu/>

8 Future Work

The AML-DL is already finished in a usable and flexible form, and two interpreters in Python and C++ exist. The language will keep evolving and being extended based on user feedback. In addition, tools for directly adding some common embedding, e.g. a semantic embedding for natural numbers, will be added for ease of use.

Regarding the communication of the AML nodes, the AML logic needs to be integrated into the AML-IP prototype. Currently the prototype is tested with DDS entities that simulate the generation of data, transmission and ingest behaviour of the AML-Engine and AML-DL nodes. Thus, since the current implementation of the AML-IP is just a prototype, the feasibility of the proposed design could have been tested.

The logic of the three main AML nodes: the AML-Engine node, the AML-DL node and the AML-CC, was already developed [1]. The extraction of their logic and adaptation to DDS, by adding the functionality for publishing and consuming topics, is the only step missing for having fully functional AML nodes.

Given the internal complexity of the AML-IP-Agent node, a more technical and low level design of the this AML-IP component will be pursued. Following a test-driven software development model, after the design and together with the implementation of the node's functionality, tests will be implemented to check the correct operation of the node.

Furthermore, this document presents a study of the tools and libraries available for the implementation of a built-in software tool for the AML-IP-Agent that exploits the UPnP protocol for automatic port forwarding on the IGD within the user's private network.

References

- [1] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Algebraic Machine Learning*. 2018. arXiv: 1803.05252.
- [2] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Finite Atomized Semilattices*. 2021. arXiv: arXiv:2102.08050.
- [3] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Semantic Embeddings in Semilattices (In preparation)*. 2021.
- [4] *patent US20190385087A1: Method for large-scale distributed machine learning using formal knowledge and training data*. 2018.
- [5] *eProsima Fast DDS Documentation - What is DDS?* Version 2.3.1. Proyectos y Sistemas de Mantenimiento SL (eProsima), 2021. URL: https://fast-dds.docs.eprosima.com/en/v2.3.1/fastdds/getting_started/definitions.html.