UNICO I+D Project
6G-DATADRIVEN-03

6G-DATADRIVEN-03-E8

Initial System Architecture

Abstract

This document describes the initial draft of the system architecture, enabling Artificial Intelligence services within future Beyond 5G and 6G networks. Furthermore, methods to exploit scenarios with distributed data sources for the training of Machine Learning models are detailed.

Financiado por la Unión Europea NextGenerationEU
GOBIERNO DE ESPAÑA MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE TELECOMUNICACIONES E INFRAESTRUCTURAS DIGITALES
Plan de Recuperación, Transformación y Resiliencia

Document properties

Document number	6G-DATADRIVEN-03-E8
Document title	Initial System Architecture
Document responsible	Carlos J. Bernardos
Document editor	Carlos Barroso Fernández (UC3M)
Editorial team	Carlos Barroso Fernández (UC3M), Carlos J. Bernardos (UC3M)
Target dissemination level	Public
Status of the document	Final
Version	1.0
Delivery date	31/12/2023
Actual delivery date	31/12/2023

Production properties

Reviewers	Antonio de la Oliva (UC3M)

Disclaimer

This document has been produced in the context of the 6G-DATADRIVEN Project. The research leading to these results has received funding from the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D programme.

Contents

List of Figures 4
List of Acronyms 5
Resumen Ejecutivo 7
Executive Summary 8
1. Introduction 9
2. AI as a Service (AIaaS) 10
3. Distributed data 11
3.1. Centralized Schemes 11
3.2. Decentralized Schemes 14
3.3. Hybrid approaches 16
4. AIaaS in 3GPP 17
4.1. AIaaS architecture 17
4.2. 5G walkthrough 18
4.3. Integration in future networks 22
5. Summary and Conclusions 28
References 30

List of Figures

Figure 1: Scheme of AIaaS....10
Figure 2: Centralized Scheme....12
Figure 3: Example of a Centralized Scheme....12
Figure 4: Example of FL....13
Figure 5: Example of a three-layer Hierarchical Scheme....14
Figure 6: Decentralized Scheme....15
Figure 7: Example of a Decentralized Scheme, or DFL....15
Figure 8: AIaaS internal description....18
Figure 9: NWDAF interfaces to expose analytics [11]....19
Figure 10: Edge enabler architecture [13]....20
Figure 11: Integrating AIaaS within 5G architecture....22
Figure 12: NRF capability exposure....22
Figure 13: Service configuration....24
Figure 14: Service configuration using an AIaaS Controller as proxy....25
Figure 15: Service workflow....26
Figure 16: Service modification....27
Figure 17: Communication via NEF....27

List of Acronyms

3GPP: 3rd Generation Partnership Project

ADRF: Analytics Data Repository Function

AI: Artificial Intelligence

AIaaS: AI as a Service

API: Application Programming Interface

B5G: Beyond 5G

CL: Centralized Learning

DCCF: Data Collection Coordination Function

DFL: Decentralized Federated Learning

DL: Decentralized Learning

DT: Digital Twin

EAS: Edge Application Server

EASDF: Edge Application Server Discovery Function

ECS: Edge Configuration Server

EDN: Edge Data Network

EEC: Edge Enabler Server

EES: Edge Enabler Client

IoT: Internet of Things

MEC: Multi-access Edge Computing

ML: Machine Learning

MFAF: Messaging Framework Adaptor Function

NF: Network Function

NRF: Network Repository Function

NWDAF: Network Data Analytics Function

O-RAN: Open Radio Access Network (RAN)

OAM: Operations, Administration and Maintenance

PDU: Packet Data Unit

PSA: PDU Session Anchor

QoS: Quality of Service

RAN: Radio Access Network

RIC: RAN Intelligence Controller

UE: User Equipment

UL: Uplink

Resumen Ejecutivo

El objetivo de este trabajo es diseñar un boceto de la arquitectura del sistema de las futuras redes B5G y 6G, centrándose en los elementos necesarios para que dichas redes puedan ofrecer servicios de Inteligencia Artificial tanto a otros elementos de la red como a usuarios finales.

El documento profundiza en el concepto de AIaaS: ofrecer servicios de IA basado en un sistema de subscripción, siendo los servicios de ML los más utilizados. Para aprovechar todo el potencial de los servicios de ML, el documento aporta una descripción detallada de los diferentes esquemas existentes en función del tipo de topología del sistema.

Se presentan los diferentes bloques funcionales que deben existir para poder ofrecer AIaaS, con las correspondientes descripciones de sus funciones e interacciones. Además, se integran dichos bloques dentro de la arquitectura presentada en los estándares del 3GPP.

Para finalizar, se aportan los diagramas de flujo de alto nivel seguidos para la configuración y provisión del servicio.

El resto del documento está redactado en inglés, de cara a maximizar el impacto del trabajo realizado en este proyecto.

Financiado por la Unión Europea NextGenerationEU
GOBIERNO DE ESPAÑA MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL SECRETARÍA DE ESTADO DE TELECOMUNICACIONES E INFRAESTRUCTURAS DIGITALES
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D

Executive Summary

The objective of this work is to design a sketch of the system architecture of the future B5G and 6G networks, focusing on the necessary elements so that these networks can offer Artificial Intelligence services both to other elements of the network and to end users.

The document delves into the concept of AIaaS: offering AI services based on a subscription procedure, with ML services being the most popular. To take advantage of the full potential of ML services, the document provides a detailed description of the different existing schemes depending on the type of system topology.

The different functional blocks that must exist to be able to offer AIaaS are presented, with the corresponding descriptions of their functions and interactions. In addition, these blocks are integrated into the architecture presented in the 3GPP standards.

Finally, the high-level flow diagrams followed for the configuration and provision of the service are provided.

1. Introduction

Nowadays everybody knows (more or less) what Artificial Intelligence (AI) is. However, there are thousands of definitions, and its boundaries are diffused. Keeping apart this problematic and polemic topic, what it is clear is that it is generating a technological revolution, helping in a lot of different fields, and day by day being more common in our lives.

AI solutions can also be implemented in networks to automatize problems. For example, the acceptance and configuration of new services or flows, forecasting the behaviour of the network in function of how the traffic evolves, or dynamically adjust the resources assigned to each end user.

From AI, one can highlight Machine Learning (ML). It is a field that uses data from the environment to adapt a model to the specific case. The process is called training because the model learns from the data. One of the big problems that ML – and, in particular, Deep Learning (DL) – has is obtaining enough data to successfully train the models. In many cases, this data is even distributed among the system, what generates problems of energy and resource consumption, and reduction of training performance (measured by the accuracy of the final model). The problem has been studied by many researchers, and it is still a hot topic nowadays. Solutions tries to reduce the resource usage and increment the accuracy of the model, but also depends on how the data is exchanged through the network.

However, this powerful tool is an unreachable mystery for many people [1]. To provide it to a broader public, AI as a Service (AIaaS) is created. It offers all the advantages of AI models and algorithms just by subscribing to it. The idea behind it is the same basis of its sisters '*as a Service*', like Software as a Service (SaaS).

New generation use cases demand ultra-low latencies, among other stringent requirements [2]. To fulfil it, networks have moved the computations as close to the end user as possible, which may reduce at the same time the energy consumed (also a milestone in next generation networks), among other solutions. AIaaS must also be adapted to these use cases, integrating it with the network elements.

The objective of this manuscript is to define a draft for the integration of AIaaS within the future Beyond 5G (B5G) or 6G networks, where not only the network functions will obtain benefits from it, but also end users as smartphones or vehicles. Additionally, it presents and describes the possibilities from the literature to exploit scenarios where the data needed for the AI service is distributed among the system. First, in the second section, the concept of AIaaS is explained in detail. In the third section, the problems of scenarios with distributed data are presented, and literature solutions are reviewed. Then, fourth section presents a possible architecture for integrating AIaaS within B5G or 6G networks. Last, section five summarizes and concludes the contributions.

2. AI as a Service (AIaaS)

As the field of AI is so complex and diverse, a high level of expertise is needed to exploit its potential at a maximum level. This is a disadvantage for its users and a constraint to its usefulness.

To solve the gap, AI as a Service (AIaaS) has arisen [3]. It provides AI services (models, data processing tools, optimized computer power, …) to other entities just by subscribing to it, without developing their own service and investing in software and hardware – see Figure 1. These other entities may be network functions to provide services as predictive network resource allocation to mitigate QoS drops; or end users, such as IoT devices and smartphones.

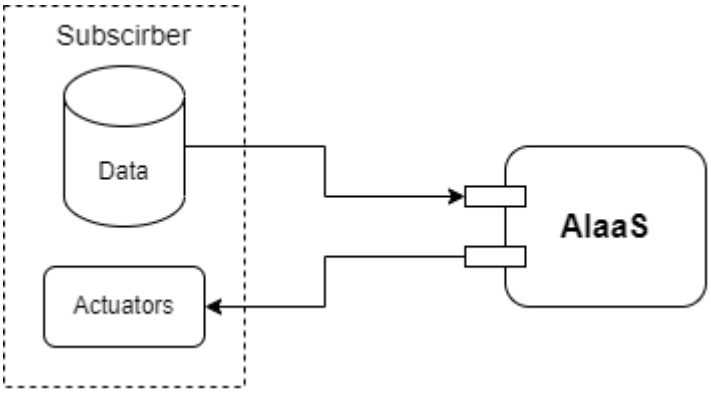

FIGURE 1: SCHEME OF AIaaS

AIaaS provide its services through two different types of subscriptions. First is used by users that want to protect their data and process the model by their own, and the other captures the opposite case, when users do not have access to enough computational resources and send their data to be processed.

Most notable and used tool from AI is Deep Learning [4], where complex models are trained with data to produce accurate results. Hence, AIaaS also allows the training of these models with data from users. Same as before, training may be performed by the user (to protect its data) and only sending model parameters, or by the resources provided by AIaaS, in case the user does not have enough computational resources. However, these models need large amounts of data to be properly trained (i.e., to produce accurate answers). To tackle this problem, one can exploit data from different sources to jointly train the model.

3. Distributed data

In real scenarios, it is typical to encounter data sources spread among the system. For example, a company with several buildings located in different places, a group of hospitals, each one with its own database, a group of IoT devices, vehicles, or smartphones, etc. There are thousands of examples and, as aforementioned, one can use all data to train highly accurate models.

However, it is not straightforward how to do it since new problems and questions come into the picture: it is clear that you want to use all data but, where do you train the model? This process is normally computationally expensive, so it needs a considerable amount of that kind of resources; you need also to care about how to transport the information, as moving such amount of data can collapse the network if it is not prepared; additionally, sources may not trust each other, then they are not willing to share their real information. And these decisions not only affect the duration or the energy consumption, but also the accuracy of the resulting model.

Fortunately, the topic is already studied by thousands of researchers [5,6], and competitive techniques are already developed and proven to deal with previous problems. Furthermore, it is still growing and a hot topic in the research area of the AI. The ideas behind these techniques are the following:

- Reduce the amount of transmitted data with compression techniques.
- Select only significant information.
- Divide the computing task among several nodes.
- Sharing only abstracted data to maintain the privacy of the information.

One way to classify them is regarding the abstracted scheme followed during the training process. In other words, the graph that captures the data exchanges, which is bounded by the topology of the system. The scheme can be centralized if exist a central coordination point, or decentralized if it does not exist. There are also some solutions that lives between the two classes, trying to get the advantages of both worlds.

3.1. Centralized Schemes

First and most basic idea when one needs to work with pieces distributed among several entities is gathering them together at some point. This is precisely the idea behind the centralized schemes: send all information to a central node where data is processed – see Figure 2. The central node also takes the role of the coordinator, giving the proper orders to the rest of the system to ensure that the training of the model is successfully completed.

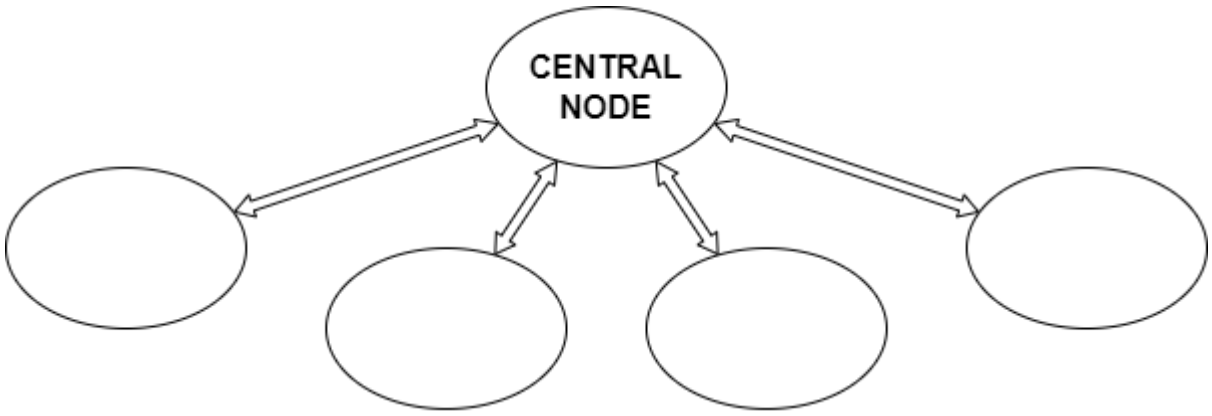

FIGURE 2: CENTRALIZED SCHEME

In case the training of the model is performed on the central node, there only exists a single instance of the model (within the central node) – see Figure 3. The information can be ciphered to a third viewer during the transmission, but not to the central node. One scenario to apply this scheme is, for example, in smart agriculture, where a bunch of sensors spread among the field send their data to a control unit, which is in charge of processing it.

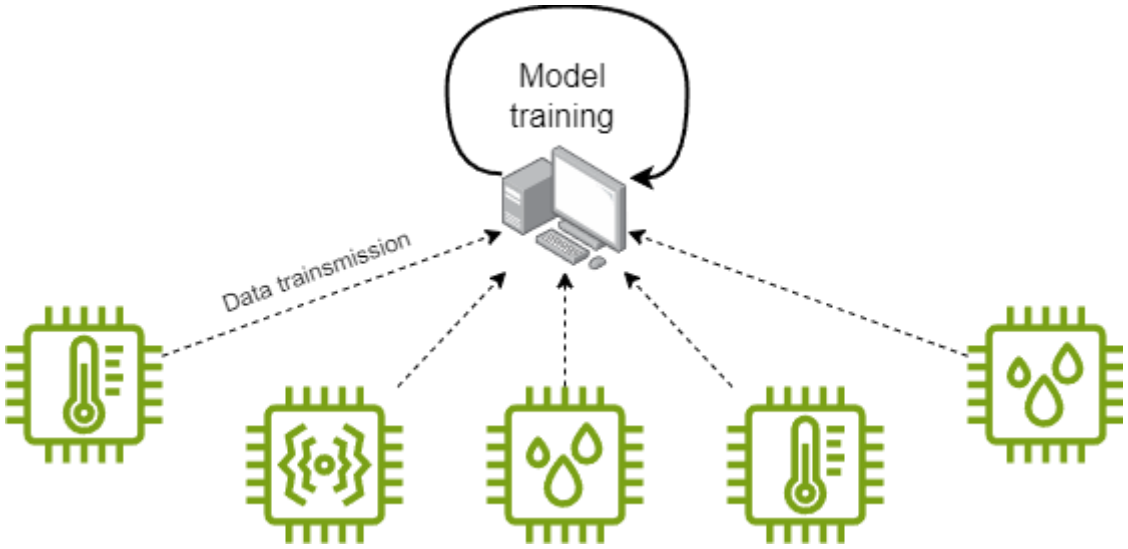

FIGURE 3: EXAMPLE OF A CENTRALIZED SCHEME

Federated Learning

Recently, a new paradigm has arisen that can be englobed also within the centralized schemes. It is called Federated Learning (FL) [7], and it is thought to maintain the privacy of the data. However, to understand its basis, we need to explain first how model training works. Deep Learning models transform the input data with a considerable number of layers, each of them is configured with a set of parameters or weights. Hence, the output depends on all these parameters, normally called model weights. The training consists in adjusting all these weights to obtain the desired output. A common way to do it is through gradient descent, aligning the model weights in a vector. Notice that two different weights vectors represent two different states of the model.

Going back to FL, what it proposes is to train local models within the sources of data and send only to the central node the resulting weights of all local models trained in the sources. Specifically, these are the steps that are performed, that can also be shown in Figure 4: (*i*) the central node, which has the model, sends a copy to all data sources; (*ii*) then, all data sources train their local models with their own data and (*iii*) send the model weights (after training) to the central node, which (*iv*) aggregates all weights into the global model. The process is repeated until some condition is satisfied, for example, the model reach a certain level of accuracy. From the steps, there is one that has not been mentioned until now, the aggregation of weights. As model weights can be represented as a vector, one common approach is to average all of them, but there also exists other ways, as weighted averaging.

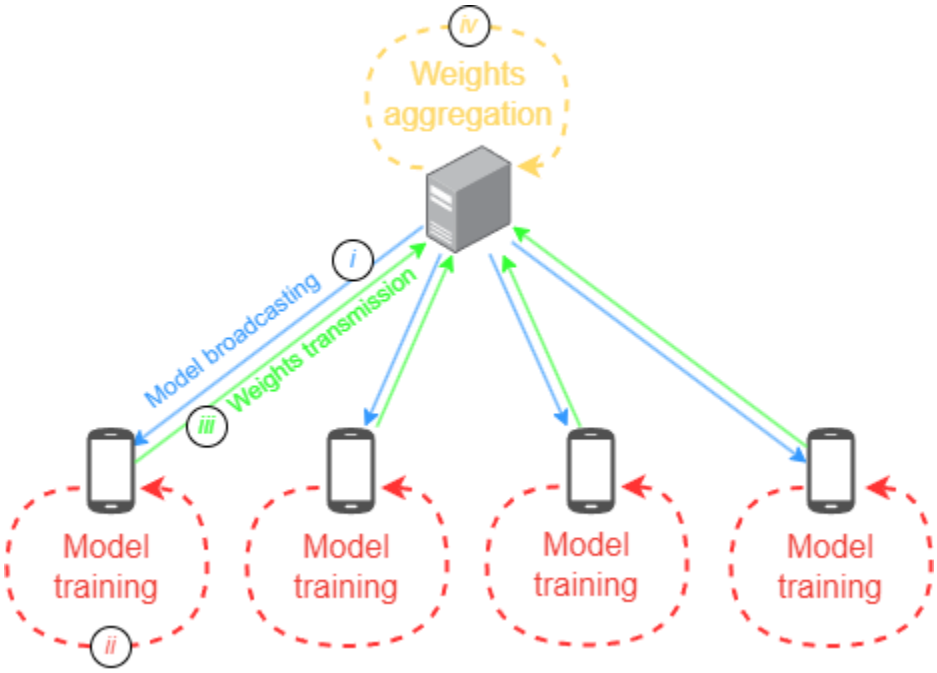

FIGURE 4: EXAMPLE OF FL

Remember that one of the pillars of FL is the fact that can preserve the privacy of the data but, how it is achieved? As the central node only receives model weights, is difficult for it to derive the raw data which generated those weights, maintaining in this way the privacy of the information. This privacy insurance makes it very attractive for a bunch of use cases, and its popularity is enormous. Nevertheless, FL also adds some restrictions: as one can see, data sources must have computational resources available to perform the local training. An example where FL can be applied is an application that uses the end devices (smartphones) to gather information and train a model.

HIERARCHICAL SCHEMES

FL did not only suppose an important step for the implementation part, but also for the research field, as opened a window of new possibilities (apart from a whole new area). Centralized schemes have evolved to more sophisticated solutions where they converge to the central node in more than one step, involving additional nodes. These solutions are called

Hierarchical schemes, each of the steps representing a layer of the hierarchy. It can also be seen as a tree graph, where the root is the central node, and the edges represent data exchanges – see Figure 5.

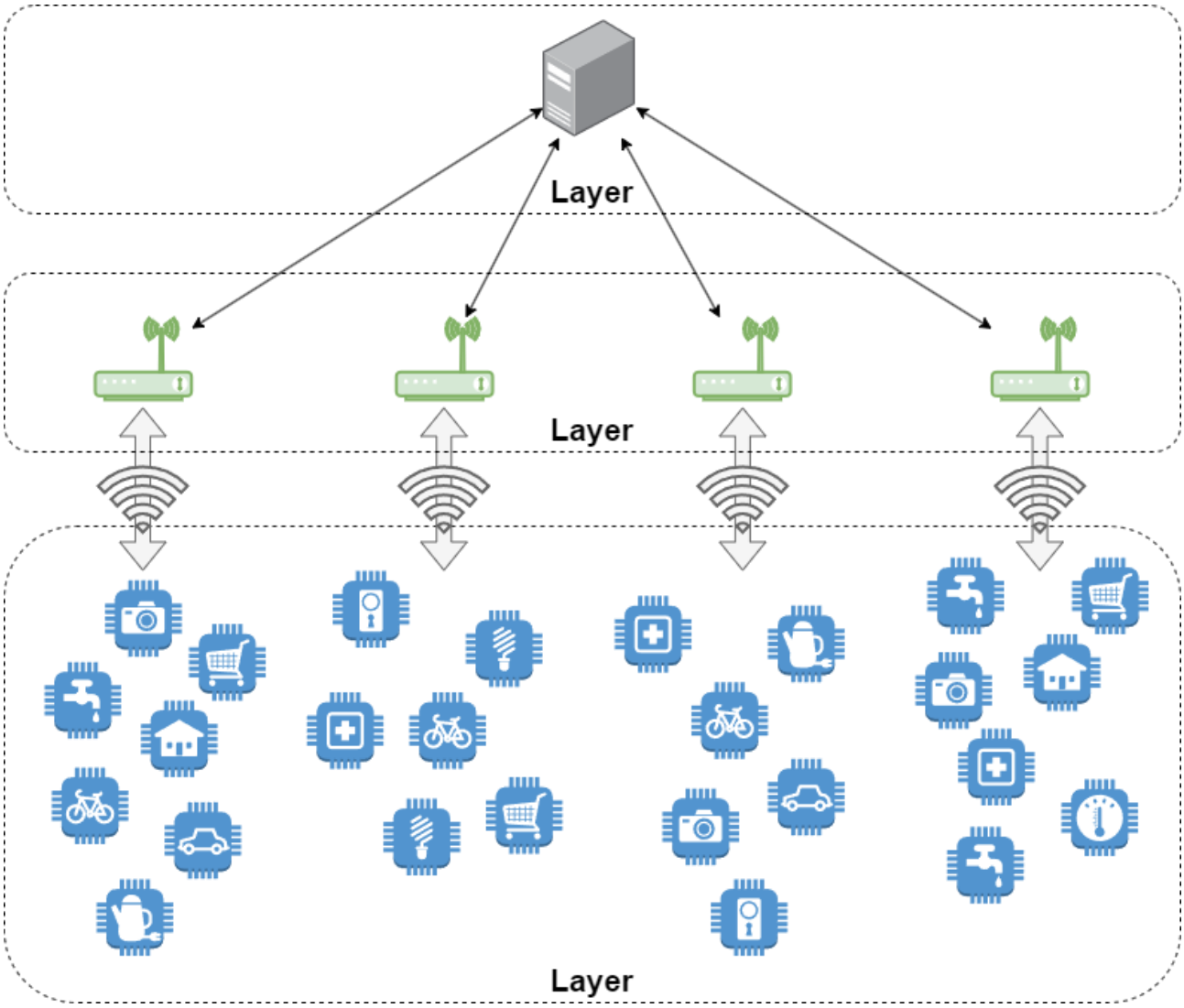

FIGURE 5: EXAMPLE OF A THREE-LAYER HIERARCHICAL SCHEME

The training of the models is normally performed in one of the two first layers, being the rest of them used for aggregating the model weights. These schemes are normally used in dense scenarios, with a large number of elements, for example, the three-layer hierarchical scheme where a bunch of IoT users connected to several access points, which train the model and send the weights to a common central node to aggregate all of them.

3.2. Decentralized Schemes

The lack of central node in some use cases, or the impossibility of select one, generates the need of a new learning scheme that can fit in them. Notice that this scenario is more common that one can

thought, for example, a set of companies that train a model together, but no one wants to give more rights to the other.

The solution, then, focuses on eliminating the figure of central node and allow nodes to exchange data among them – see Figure 6. In this case, each node has its own instance of the model, which learns from the local data and the receiving information. However, over an enough number of rounds, all models converge to the same point.

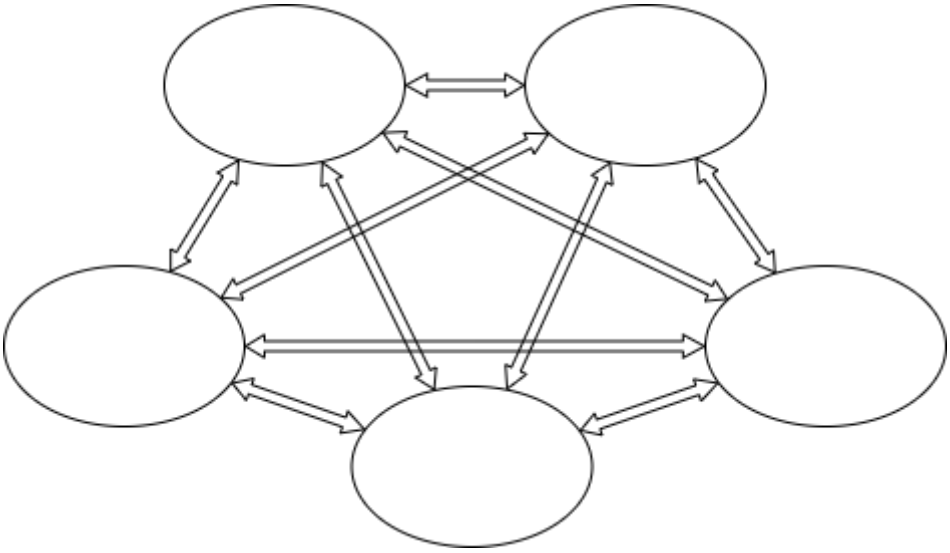

FIGURE 6: DECENTRALIZED SCHEME

FL also influenced the decentralized schemes. The idea is that all nodes perform the training and aggregation of weights, as explained in the following: the process starts with the assumption that all nodes already own an instance of the same model; first, (*i*) nodes locally train their models and (*ii*) send the model weights; then, (*iii*) also all nodes aggregate the receiving weights with their own, and the process starts again. These steps are detailed in Figure 7.

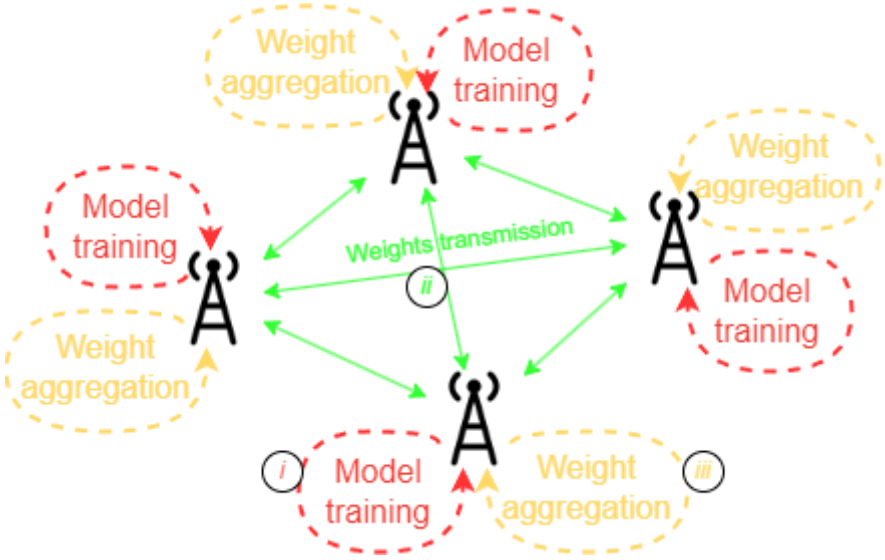

FIGURE 7: EXAMPLE OF A DECENTRALIZED SCHEME, OR DFL

Some authors call this kind of schemes Decentralized Federated Learning (DFL), but others just refer to them as Decentralized Schemes or Decentralized Learning (DL). One example of this is the cooperative training of a model among a set of Access Points.

3.3. Hybrid approaches

Some solutions cannot be categorized within any of the previous two schemes, because try to merge both of them and get advantages from both sides. There is not a common approach on how to call them. Some authors refer to Semi Decentralized Schemes, others just englobe them into the Decentralized Schemes. The characteristics of these approaches depends on how they are constructed, in function of the pieces that collect from the former schemes, and they must be studied for each case in particular.

For example, in [8] the authors propose a scheme to train models over a scenario with a set of equal nodes. This scheme is very similar to FL, where nodes send their weights to the central node. However, the central node changes between rounds, and any node of the system can be selected as the new central node.

Other example is the work [9], in this case, thought to be applied on a scenario with multiple edge servers, each of them with a set of connected devices. The presented scheme has two layers, one centralized and the other decentralized. First layer englobes the interactions between the edge servers and their connected devices, and it employs the same rules than FL to obtain a model for each of the edge servers. Second layer is the interactions among the edge servers, which exchange their models in a decentralized manner.

4. AIaaS in 3GPP

New use cases tackled in future 6G networks tends to have stringent latency requirements [2], so it is fundamental that services in those use cases fulfil the QoS. The paradigm of Edge Computing tries to solve that problem by bringing closer the applications to the end users, or specifically, to the Edge of the network. Following this line of thought, to increase the capabilities of the AI services and reduce their response times, one can bring them as close as possible to the consumer. With this development, critical actions as the dynamic allocation of resources can be solved using AIaaS solutions.

Integrating AIaaS within 6G networks will leverage its functionality and usability. A key enhancement must be the availability of these services for both end users and network elements through well-defined interfaces.

In the following subsections, we present a reference architecture for AIaaS, explaining the functionality of all existing elements, to later provide a possible integration within future networks, explaining the elements from the standard that take a role in the process.

4.1. AIaaS architecture

Figure 8 depicts the proposed architecture for a AIaaS element, and it is divided in:

- Input modification: acts as the entrance gate. Its main role is to prepare the data to be processed by the computing node, which results indispensable when treating AI solutions because data in the wrong format may corrupt the whole model.
- Output translation: opposite to the former entity, this is the exit gate. Its function is to translate the result from the AI algorithms or models to one understandable by the current agent.
- AIaaS Worker: is responsible for executing the steps from the AI algorithm, process the data through the ML model, or aggregate the weights from agent models. Basically, it consumes the data from the agent to produce the answer given by the selected AI service.
- AIaaS Controller: is the brain of the subsystem and configures the rest of elements to provide the service accordingly to the requirements of the task at hand. In particular, it sends the AI algorithm or model that satisfies the selected service to the Computing node, specifying all the required actions to fulfil the task; and provide the format of the data to the Input modification and the Output translation. One instance of the controller may service to one or more instances of the rest of the components.
- AIaaS Agent: represents a subscripted user to the service, which sends its data and receives the output. As aforementioned, it can use the computing resources from the service or not, to protect the privacy of its information.

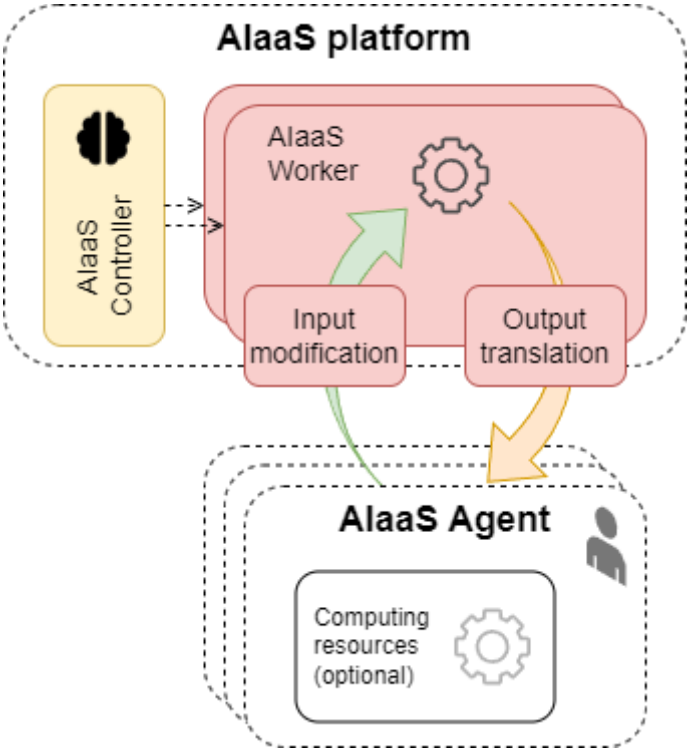

FIGURE 8: AIaaS INTERNAL DESCRIPTION

Green arrows in Figure 8 represents the data flow from the agent to the computing node where this data is processed. By contrast, orange arrow exemplifies the answer given by the service to the agent, which also encompasses the transmission of model weights.

The controller has two additional interactions, apart from those described in Figure 8. By one side, an external entity must communicate the supported services by the AIaaS entity to the controller when it is being implementing, or if a new service will be provided. This entity may be the network control plane, or directly, the administrators of the network. By the other side, the controller can communicate with other AIaaS controllers that provide the same service. This allows to leverage the distributed schemes explained in former section to increase the performance of the employed ML model, using data from other domains.

Notice that AIaaS may satisfy several different services at the same time. For this purpose, it may be configured with all the appropriate algorithms and models.

4.2. 5G walkthrough

Before getting to work in the integration, we describe some NFs, interactions, and procedures from 3GPP standards that can result useful and are related to AIaaS.

3GPP defines in [10] and detail in [11] the Network Data Analytics Function (NWDAF). It supports the collection and analysis of data and exposes the results to other elements of the network. Its objective is to provide the needed tools to the rest of the Network Functions (NFs), and even the Operations, Administration and Maintenance (OAM) to take decisions or actions leveraging data

and its analysis. This analysis of the data may be performed using ML models, also allowing the training of them.

The functionality of the NWDAF can be complemented or split using the following optional NFs:

- Data Collection Coordination Function (DCCF): coordinates the collection of data, instructing the proper NFs to provide it. It can also format and process the data and redistribute to the consumer. This functionality can be covered by the proper NWDAF.
- Analytics Data Repository Function (ADRF): serves as a database for all kind of information: raw data exposed by the sources, results generated by the NWDAF, and the trained models that are used.
- Messaging Framework Adaptor Function (MFAF): processes, formats, and sends data to the consumers following a messaging framework.

To visualize the interactions among these NFs, Figure 9 from [11] depict interfaces among them, in this case, to expose the analytics from the NWDAF. Notice that the data collection procedure is analogue, and the ADRF can substitute any side.

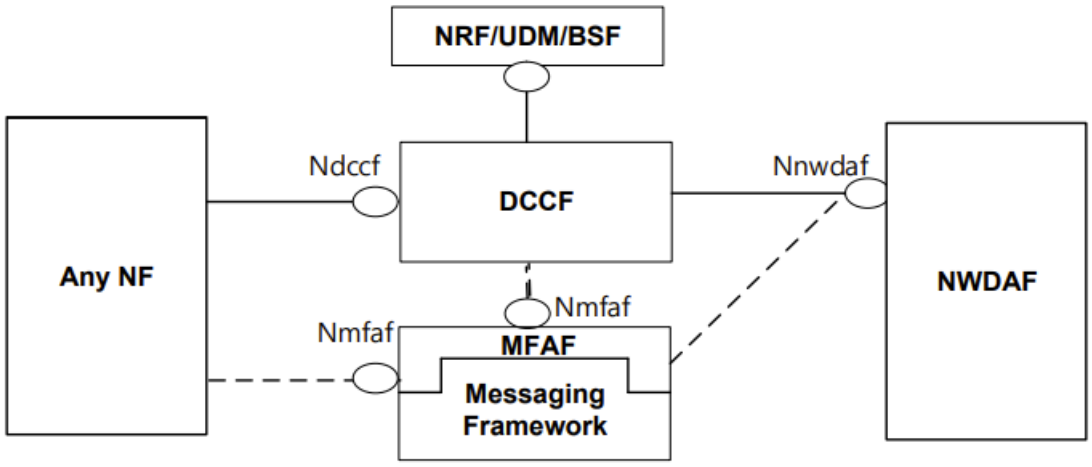

FIGURE 9: NWDAF INTERFACES TO EXPOSE ANALYTICS [11]

The standard also supports the deployment of more than one instance of the NWDAF, which can be specialized in different tasks or the same (leveraging scenarios with distributed data sources with FL).

3GPP standards also define in [12] the requirements and architecture necessary to provide Edge Computing services. They offer interfaces and Application Programming Interfaces (APIs) tailored for application developers to harness edge capabilities – see Figure 10 from [13]. Developers gain the ability to discover, establish connections with, and seamlessly transition between various application servers situated on the edge network. This access empowers them to maximize the potential of the underlying 3GPP network, enhancing and optimizing their services by leveraging edge computing capabilities.

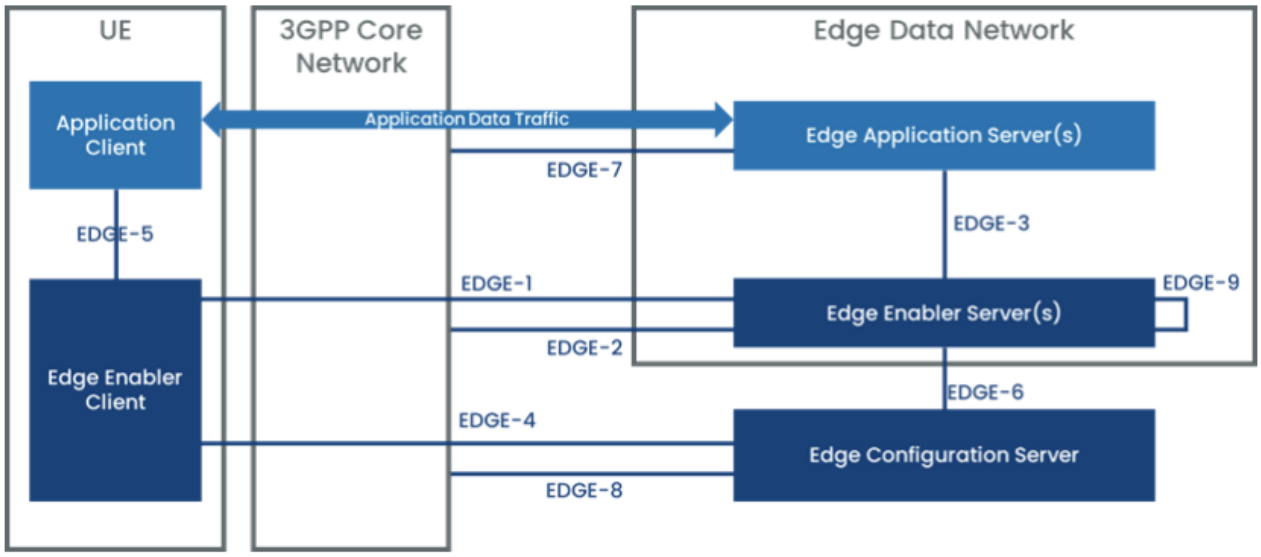

FIGURE 10: EDGE ENABLER ARCHITECTURE [13]

In the following, all functional blocks represented in Figure 10 are defined one by one, highlighting the relationship and interactions between them.

Application Client (AC)

The AC, or Application Client, operates within the User Equipment (UE) and functions as the client for applications.

Edge Application Server (EAS)

EAS functions as the server for applications. ACs connect to the EAS to access application services leveraging Edge Computing advantages. While some application server functions might exclusively be available as an EAS, others might exist both at the edge (EAS) and in the cloud. These functions might be identical or differ between the EAS and cloud-based counterparts, potentially leading to variations in the exchanged Application Data Traffic with the AC.

The EAS can interact with the 3GPP Core Network capabilities in several optional ways. It can access 3GPP Core Network capabilities via the edge enabler layer through the EES, or it can directly access 3GPP Core NFs if is trusted (using capability exposure functions like NEF).

Edge Enabler Server (EES)

EES plays a crucial role in facilitating functionalities essential for EASs and EECs. Its key responsibilities include provisioning configuration data to EEC, offering API invocation and exposure functions, to allow the exchange of application data traffic with the EAS. It also can interact with the 3GPP Core Network, supporting external exposure of network and service capabilities.

Edge Enabler Client (EEC)

It provides the retrieval of configuration parameters by the AC to enable the exchange of application data traffic between the application located in the edge server, and is in charge of discover available EASs and control the mobility of the UE, among other configuration parameters.

Edge Configuration Server (ECS)

ECS serves as a critical support system enabling connectivity between EECs and EESs. Its functionalities encompass provisioning essential edge configuration information to EECs, including details to distinguish among EESs and establishing connection parameters.

Furthermore, ECS supports registration functions for EESs, API-related operations, and interfaces with the 3GPP Core Network to access network capabilities. It manages service provisioning information with partners and retrieves data from them as well. ECS can also facilitate ECS discovery for unconfigured EAS by utilizing Edge repository functions.

Additionally, ECS-ER, an enhanced version, expands capabilities to support federation and common EAS discovery by registering, storing, and exchanging edge computing resource data among ECSs and ECS-ERs within the federation, providing a centralized repository for common EES and EAS information.

Additionally, the 5G network incorporates diverse functionalities and enhance others to facilitate edge computing, such as traffic steering, local area network connectivity, and the ability to support three distinct connectivity models delineated in [14].

- Distributed Anchor Point: Utilizes a local Packet Data Unit (PDU) Session Anchor (PSA) User Plane Function (UPF) to direct all UE traffic to the local site, occasionally optimizing routing via re-anchoring.
- Session Breakout: Employs a central site's PSA UPF along with one or more local site PSA UPFs for a PDU session. The edge computing application traffic is selectively directed to the local PSA UPF using mechanisms like Uplink (UL) Classifier or multi-homing Branching Point.
- Multiple PDU Sessions: Involves using PDU sessions with local and central site PSA UPFs for various applications. The local PSA UPF may change due to UE mobility or other factors.

The 5G core network also supports the Edge Application Server Discovery Function (EASDF), enabling the discovery of relevant EAS IP addresses for edge applications. This feature assists in identifying EASs during application initiation or when selecting a new EAS for specific functionalities, acting as a Domain Name System (DNS) resolver. For fulfilling its obligations, EASDF has an interface to communicate with EASs through the UPF.

4.3. Integration in future networks

Regarding the previous discussion, it is clear that the NWDAF can provide AI services to other NFs, but it is restricted to provide network analytics and lacks the provision to the UE. To fulfil the gap, Edge Computing can be leveraged.

Figure 11 presents the architecture integrating AIaaS functional blocks from Figure 8 into the 5G network architecture and depict the interactions between them. As one can see, the AIaaS Controller acts as a new NF from the viewpoint of the 5G core, and the AIaaS Workers are deployed as NWDAF instances or applications in the Edge, in function of whether the service that it will fulfil needs data from the user or not.

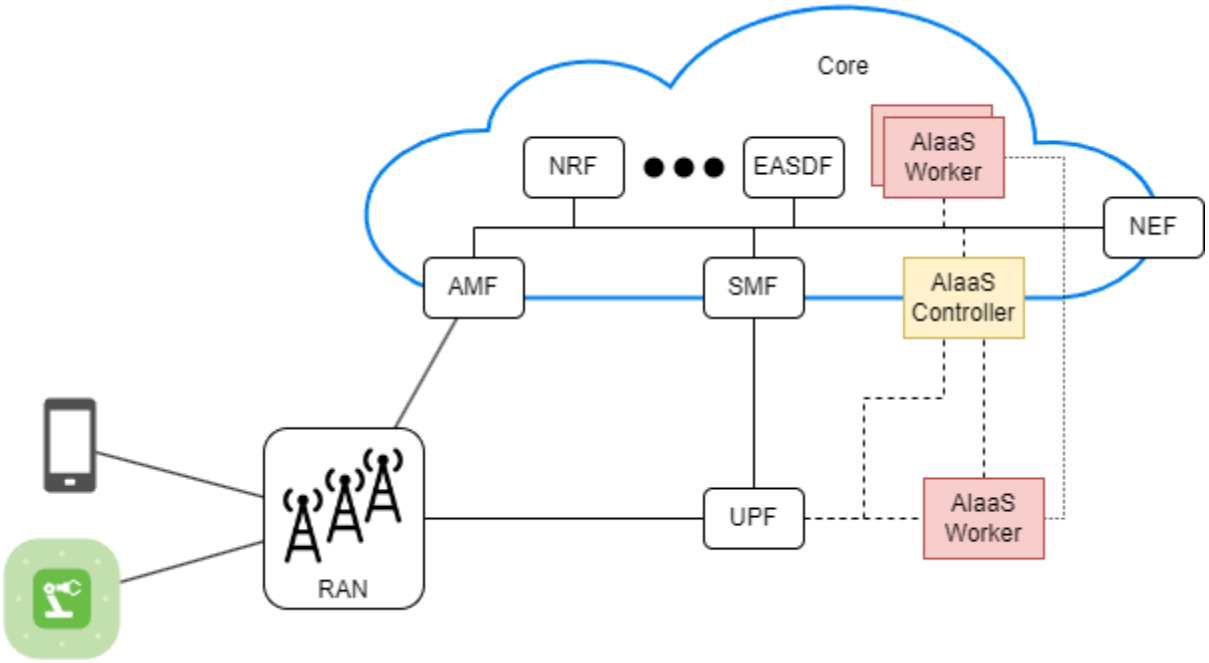

FIGURE 11: INTEGRATING AIaaS WITHIN 5G ARCHITECTURE

The AIaaS Controller needs to register in the Network Repository Function (NRF) itself and all the services that it is able to provide, exposing them to the rest of NFs. Also, it can use the NRF services to discover other AIaaS Controllers and their services. All these interactions are carried out through the Nnrf interface.

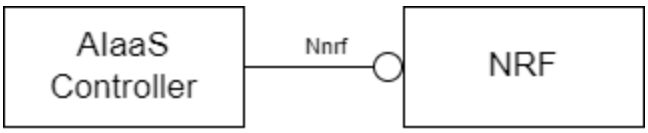

FIGURE 12: NRF CAPABILITY EXPOSURE

With this procedure, any NF is aware of the services that the AIaaS Controller can provide, and how can reach it. To provide this capability also to the UEs, the functionality of the EASDF can be leveraged. As it acts as a proxy between the UE and the DNS, it is straightforward to expand its

functionality to provide also the address of the AIaaS Controller when UE want to access to its services.

AIaaS Workers are volatiles, i.e., they are instantiated and destroyed, by the AIaaS Controller, on demand. Then, when any entity wants to ask the AIaaS Controller for a service, the process that occurs is the following, described in Figure 13:

i. First, the client contacts the AIaaS Controller to indicate the needed service, establish the characteristics (e.g, if the client will execute model training by its own to not expose its private information) and negotiate its QoS requirements (e.g., latency, or minimum accuracy for the ML model).
ii. The AIaaS Controller determines the AIaaS Workers that are needed to satisfy the promised service. As the figure shows, AIaaS Workers can be located in the core, as an NWDAF, to offer its services to the rest of NFs, or in the Edge Data Network (EDN), as an edge application within the EAS, to provide services to applications from the UE side. Some scenarios may require more than one AIaaS Worker to successfully provide the service, so the AIaaS Controller must also decide the interactions among all the elements, workers and agents.
iii. Once everything is decided, the AIaaS Controller must instantiate the AIaaS Workers, providing them the AI model or algorithm that will consume the data, the data structure that will receive and the type of output that must be sent to the AIaaS Agent. Additionally, the AIaaS Controller must provide the interfaces that the AIaaS Worker will need to exchange information with other workers and the instructions to do so, in function of the decided scheme.
iv. While step iii is being performed by the AIaaS Controller, AIaaS Agents must be also instantiated. As the AIaaS Workers, the AIaaS Agents has also different possible locations, in function of where the data sources are connected to. In case the source is a NF, the AIaaS Controller is the one in charge of configuring it.
v. Finally, the AIaaS Controller communicates to the client that the platform is instantiated correctly (or not, in case of some failure).

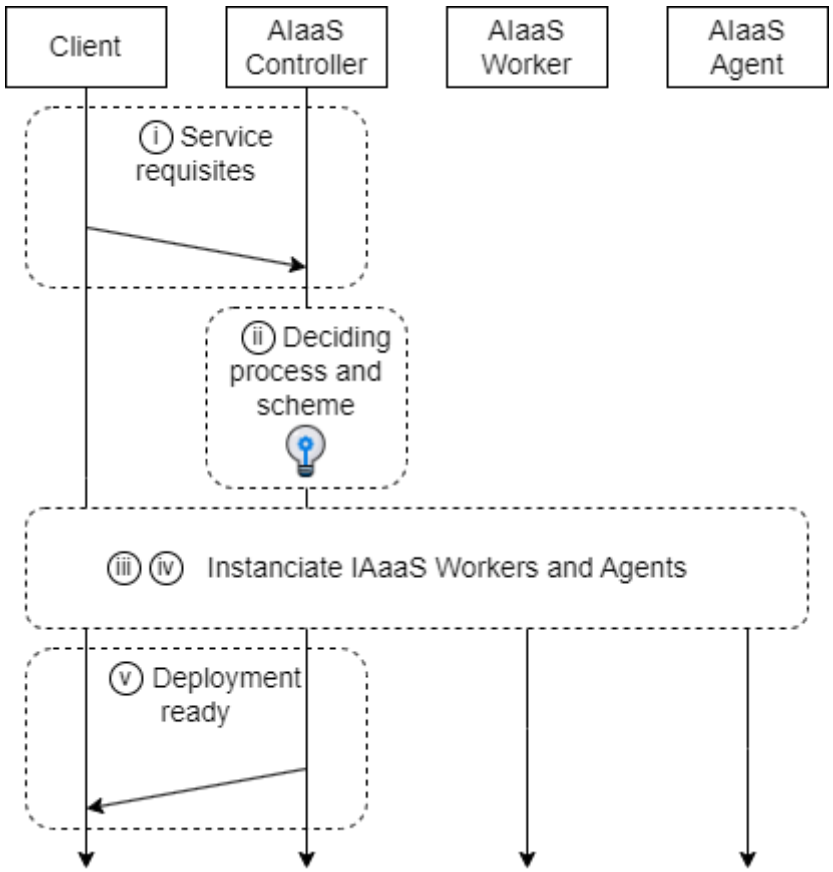

FIGURE 13: SERVICE CONFIGURATION

In case that the AIaaS Controller is not capable of satisfying all requisites of the needed service, it can ask other AIaaS Controllers to expand its capabilities, or directly act as a proxy between the consumer and the other AIaaS Controller, as is depicted in Figure 14.

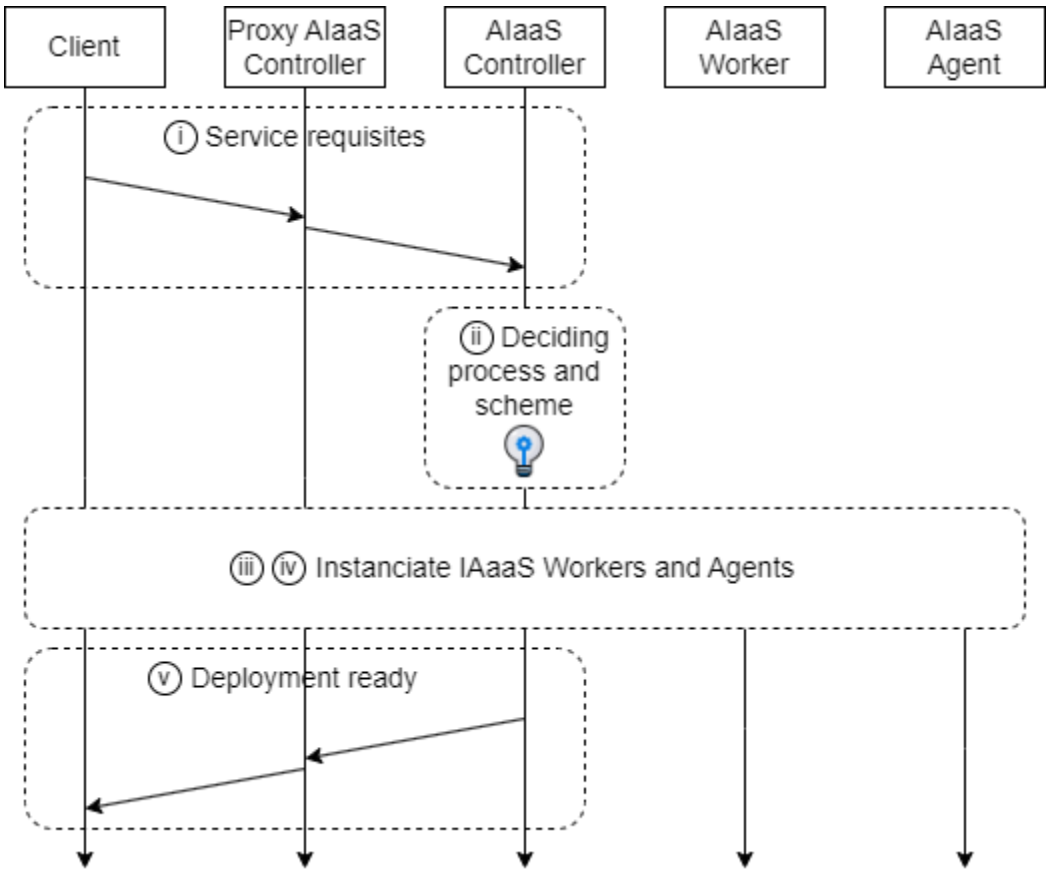

FIGURE 14: SERVICE CONFIGURATION USING AN AIaaS CONTROLLER AS PROXY

After instantiating all elements and configuring the interactions, the service can begin to be satisfied. The process, in this case, will depend on the scheme proposed by the AIaaS Controller, but is generally involves the next steps:

i. AIaaS Agent that has the data source preprocess the data following the given instructions. Under specific scenarios, it will also train the AI model. This step is optional and will be implemented only if the situation needs it.
ii. The data is transmitted from the AIaaS Agent to the Input Modification point of the AIaaS Worker, which modifies (if needed) and deliver it to the worker.
iii. The AIaaS Worker that receives the data will perform the actions the AIaaS Controller has indicated in its creation. This englobes the execution of algorithms, model training or inference, model weights aggregation, etc.
iv. Last, the AIaaS Worker sends the processed answer through the Output Translation point to the AIaaS Agent that consumes the answer. Notice that the consumer may be different from the source if the client of the service indicates it.

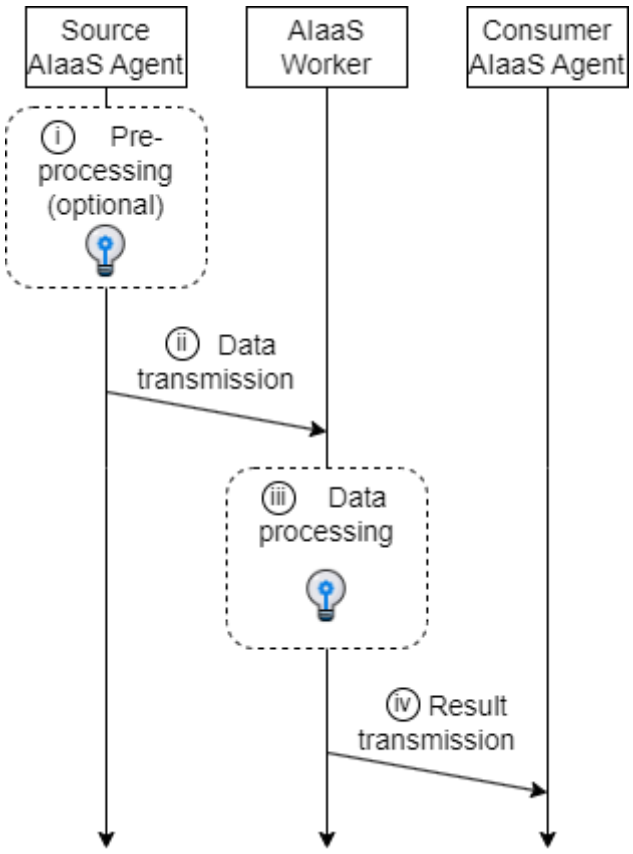

FIGURE 15: SERVICE WORKFLOW

When step iv is finalised, the process may be repeated continuously until the service is satisfied. During this time, the AIaaS Controller may desire to change some parameters of the process, either for a direct request of the client, e.g. by updating the QoS requirements, or due to a change in the environment that alters the service provisioning. In this case, the AIaaS Controller will send the new configuration to all elements that are affected and notify the client, if necessary, as one can see in Figure 15.

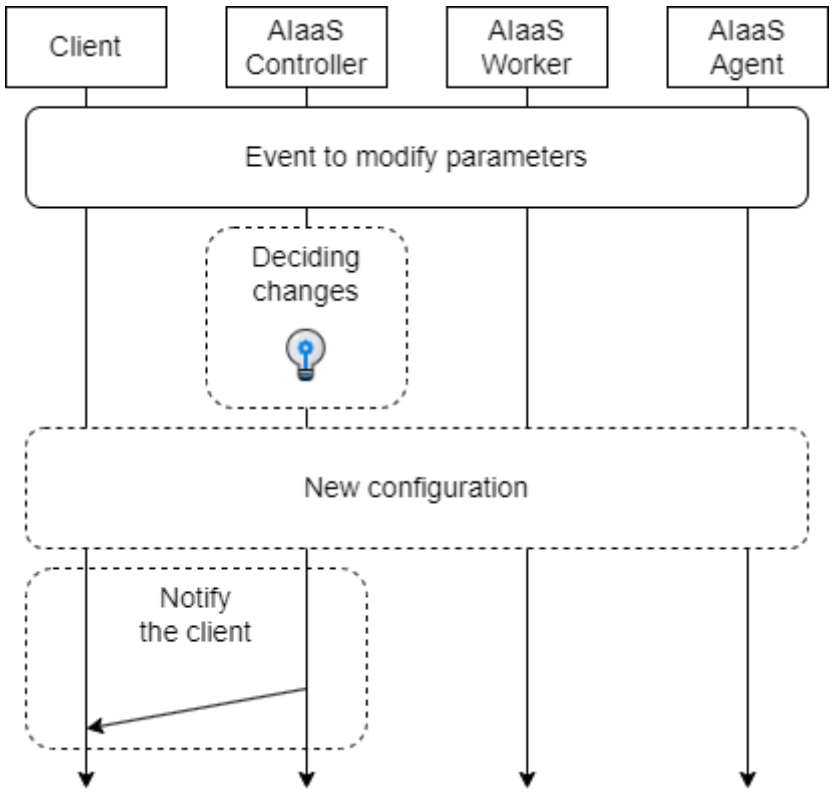

FIGURE 16: SERVICE MODIFICATION.

Once the service is complete and correctly satisfied, the AIaaS Controller may eliminate the elements that are not needed anymore to release the resources.

Remark that all these aforementioned interactions are described under the condition that all entities are reliable and trustworthy. In any other scenario, the untrusty element must interact with the NEF, using Nnef interface. Then, NEF will interact with the proper trusty NF to retrieve the needed information.

FIGURE 17: COMMUNICATION VIA NEF

5. Summary and Conclusions

In exploring AIaaS, it becomes apparent that leveraging AI's full potential demands a high level of expertise. AIaaS addresses this expertise gap by providing AI services to entities without the need for extensive infrastructure development. This service includes AI models, data processing tools, and optimized computing power. It caters to various entities, including NFs from the 5G core and end users like IoT devices and smartphones.

AIaaS operates through two subscription types, accommodating users who want to protect their data by processing models themselves or those lacking computational resources who opt to send their data for processing. Deep Learning, a significant AI tool, plays a key role in AIaaS, enabling the training of complex models for accurate results. However, training these models requires substantial data, often sourced from distributed data pools.

The challenge arises with distributed data sources. Real-world scenarios present data sources spread across various entities like companies with multiple locations, hospitals with distinct databases, and IoT devices. Combining data from these sources poses several challenges:

- Data Training Locations: Determining where to train the model, considering computational expenses and network limitations.
- Data Transmission: Moving substantial data amounts without overburdening the network.
- Data Privacy and Trust: Ensuring data privacy as sources may not trust each other, hindering information sharing.

Researchers have developed competitive techniques to address these challenges. These solutions aim to reduce transmitted data using compression techniques, select significant information, distribute computing tasks across nodes, and abstract data to maintain privacy.

The document presents the two primary categories, centralized and decentralized schemes, which emerge to address these data challenges:

- Centralized Schemes: Concentrate data at a central node for processing and model training.
- Decentralized Schemes: Eliminate the central node, allowing nodes to exchange data among themselves for training and aggregation of weights.

Federated Learning (FL) introduces a new paradigm within centralized schemes, aiming to maintain data privacy. FL entails training local models within data sources and sending resultant weights to a central node for aggregation. This method preserves data privacy by only transmitting model weights, making it attractive for various applications.

Hierarchical schemes evolve centralized solutions by introducing additional nodes, forming multiple layers of data exchange. These schemes find applications in dense scenarios with numerous elements, aiding in aggregating model weights.

AIaaS's integration within 6G networks holds significant promise, exploiting all the power of AI services leveraging the high performance of next generation networks, as it already does the

framework of Edge Computing. This integration requires well-defined interfaces for AI services, catering to both end users and network elements.

This document proposes an AIaaS reference architecture that includes the following components: Input Modification, Output Translation, AIaaS Workers, AIaaS Controller, and AIaaS Agents. This architecture facilitates data flow, processing, and service provision within the AIaaS framework.

Within 3GPP standards, the Network Data Analytics Function (NWDAF) supports data collection and analysis, allowing decision-making based on analyzed data. The 5G architecture facilitates Edge Computing, enabling connectivity between Edge Enablers, Application Servers, and the 3GPP Core Network.

The integration of AIaaS within future networks involves leveraging NWDAF functionalities and Edge Computing to bridge gaps in service provision to end users. The presented architecture shows interactions between AIaaS elements and their incorporation into 5G network architectures. It's powered by the AIaaS Controller acting as a new core entity, orchestrating AIaaS Workers strategically.

The AIaaS Controller registers services in the Network Repository Function (NRF), enabling seamless access across network functions (NFs) via the Nnrf interface. This extends services to users through the Edge Application Server Discovery Function (EASDF).

AIaaS Workers are dynamic, spawned and terminated on demand by the AIaaS Controller. Highlights of the service requests are:

- Clients engage the AIaaS Controller, specifying service needs and QoS.
- The Controller deploys AIaaS Workers, deciding their location based on data requirements—core or Edge.
- AIaaS Agents, if needed, are configured by the Controller.

The process to provide the service involves data preprocessing, transmission, AI model execution, and result delivery, managed by the AIaaS Controller.

This setup allows dynamic adaptations and parameter changes by the AIaaS Controller, concluding with resource release after successful service completion. In uncertain trust scenarios, untrusted elements interact via the NEF-Nnef interface to engage trusted NFs for information retrieval.

The complex nature of AIaaS integration requires robust interfaces, reliable interactions, and adaptability to diverse scenarios for seamless service provision within future networks.

References

[1] Castelvecchi, D. (2016). Can we open the black box of AI?. Nature News, 538(7623), 20.

[2] Banafaa, M., Shayea, I., Din, J., Azmi, M. H., Alashbi, A., Daradkeh, Y. I., & Alhammadi, A. (2023). 6G mobile communication technology: Requirements, targets, applications, challenges, advantages, and opportunities. Alexandria Engineering Journal, 64, 245-274.

[3] DATADRIVEN-03-E7 "Estudio de las necesidades y requisitos de la industria conectada en relación con la aplicación de soluciones AIaaS y sus posibles demostradores".

[4] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-6, doi: 10.1109/ICCUBEA.2018.8697857.

[5] Peteiro-Barral, D., Guijarro-Berdiñas, B. A survey of methods for distributed machine learning. Prog Artif Intell 2, 1–11 (2013). https://doi.org/10.1007/s13748-012-0035-5.

[6] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. 2020. A Survey on Distributed Machine Learning. ACM Comput. Surv. 53, 2, Article 30 (March 2021), 33 pages. https://doi.org/10.1145/3377454.

[7] H. Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. https://research.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[8] Hadeel Abd El-Kareem Abd El-Moaty Saleh, Ana Fernández Vilas, Manuel Fernández-Veiga, Yasser El-Sonbaty, and Nashwa El-Bendary. 2022. Using Decentralized Aggregation for Federated Learning with Differential Privacy. In Proceedings of the 19th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN '22). Association for Computing Machinery, New York, NY, USA, 33–39. https://doi.org/10.1145/3551663.3558682.

[9] Sun, Y., Shao, J., Mao, Y., Wang, J. H., & Zhang, J. (2022, April). Semi-decentralized federated edge learning for fast convergence on non-IID data. In 2022 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1898-1903). IEEE.

[10] 3GPP, System architecture for the 5G System (5GS), Service and System Aspects working group, Technical Specification (TS) 23.501, Release 17.

[11] 3GPP, Architecture enhancements for 5G System (5GS) to support network data analytics services, Service and System Aspects working group, Technical Specification (TS) 23.288, Release 18.

[12] 3GPP, Architecture for enabling Edge Applications, Service and System Aspects working group, Technical Specification (TS) 23.558, Release 18.

[13] Dongwook Kim, June 2023, Edge Computing, 3GPP, accessed 22/12/2023 https://www.3gpp.org/technologies/edge-computing.

[14] 3GPP, 5G System Enhancements for Edge Computing; Stage 2, Service and System Aspects working group, Technical Specification (TS) 23.548, Release 18.