**3GPP TSG-SA WG4 Meeting #128S4-240966**

**Jeju Korea, 20th -24th May 2024**

**Source: HUAWEI**

**Title: [FS\_AI4Media] IMS architecture and call flow mapping**

**Spec: FS\_AI4Media Functional PD v1.2.0**

**Agenda item: 9.6**

**Document for: Agreement**

**1. Introduction**

An update to the mapping IMS architecture and call flow.

**2. Reason for Change**

The AI/ML use cases for IMS-based real-time communication have been defined in clause 4.1.2 and 4.4.1 of the FS\_AI4Media Functional PD, this contribution proposes to make the following changes:

1) Move clause 5.3.1 and 5.3.2 to 5.2.2 and 5.2.3, which are common for both 5GMS and IMS. Meanwhile, remove the AI capability manager function defined in the original clause 5.3.2 according to S4-240782 which was agreed in last SA4 #127-bis-e meeting.

2) Add clause 5.4 for the mapping of IMS architecture and call flow for the use cases mentioned above.

**3. Proposal**

It is proposed to agree the following changes to FS\_AI4Media Functional PD v1.2.0.

\* \* \* First Change \* \* \* \*

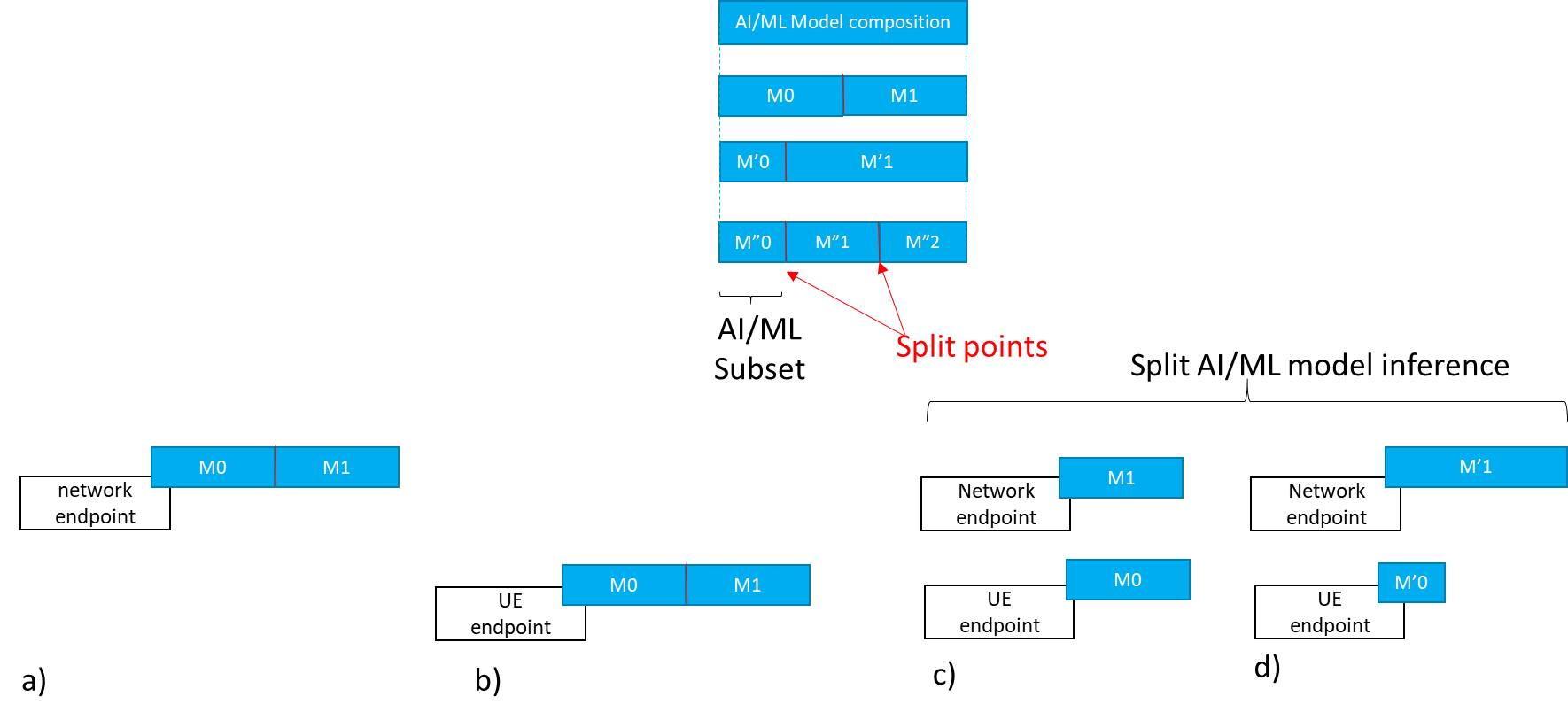
5 Media service architecture for AI/ML

5.1 AI/ML model composition

Figure 5.1-1 depicts an AI/ML model composed of different AI/ML subsets based on various split points. Several compositions of the same AI/ML model are represented by the AI/ML subsets (M0, M1), (M’0, M’1), or (M “0, M “1, M “2) with split points highlighted in red. The same AI/ML subset may be used in different compositions depending on the configurations of the model composition (e.g. M’0 and M ’00 according to figure 5.1-1).

In figure 5.1-1, (a) and (b) are examples of AI/ML inference endpoints running an AI/ML model M composed of two subsets M0, M1. A endpoint (network/UE) may run the AI/ML model subset M0 while downloading the other subset M1.

Examples (c) and (d) demonstrate AI/ML split models where M0, M’0 run on the UE while M1, M1’ run on the network respectively.



**Figure 5.1-1 AI/ML model composition example**

5.1.1 Split AI/ML model inference topologies

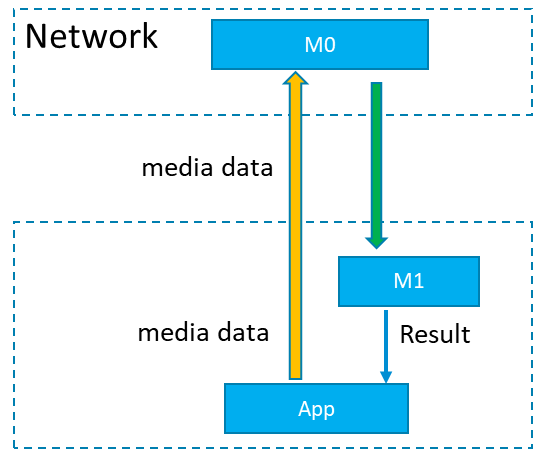
5.1.1.1 UE as media data source

Figure 5.1.1.1-1 depicts an example of split AI/ML model inference topology where the UE is the media data source and runs the first model subset M0 as described in scenario (a) of clause 4.1 (object recognition). Figure 5.1.1.1-2 depicts another example of a split AI/ML model inference topology where the UE is also the media data source but the network server runs the first subset M0 as described in scenario (b) of clause 4.1. Assuming that the necessary AI/ML model subsets are already available at each endpoint, figure 5.1.1.1-1 and figure 5.1.1.1-2 show the data exchanged between the different split inference endpoints, including input media data, intermediate data, and inference results.

The results can be a textual indication of the recognized object, an output score, a bounding box, enhanced media data, etc.



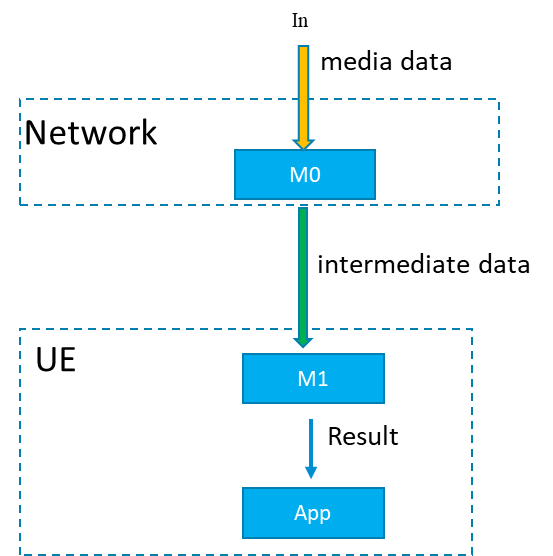
**Figure 5.1.1.1-1: Split AI/ML model inference where the UE is the media data source with first inference endpoint on the UE**



**Figure 5.1.1.1-2: Split AI/ML model inference where the UE is the media data source with first inference endpoint on the network**

5.1.1.2 Provider/network as media data source

Figure 5.1.1.2-1 depicts examples of split model topologies where the network or the AI/ML provider ingests the media data, such as in the use-case of clause 4.2 (video quality enhancement).



**Figure 5.1.1.2-1: Split AI/ML Model inference where the network/ AI/ML service provider ingests the media data**

5.2 Basic architectures and workflows

5.2.1 Introduction

Considering the related use cases as documented in TR 22.874 and also as documented in the latest version of the Permanent Document, we can start from some basic scenarios for consideration of a basic architecture for AI/ML media services.

The basic starting scenarios are:

1. Delivery of a pre-trained AI/ML model from network to UE, typically at the start of an AI media service, but may also require updates during the service. At the most basic level AI/ML models can be delivered as a file (e.g. TensorFlow SavedModel, PDF5, ONNX file, NNEF file etc.) containing all the necessary information required for the UE to perform on device inference using the delivered model. For split scenarios, a (partial) AI model to be used in the UE may be delivered.
2. Split inference of a pre-trained AI/ML model(s) with two further sub-scenarios:
   1. Basic scenario with an inference in the network or in the UE.
   2. Split scenario with inferences between the network and the UE, where the intermediate data output from the network inference (resp. UE inference) is transferred to the UE (resp. network) to be used as the input for UE device inference (resp. network inference). Depending on the characteristics of the intermediate data, such as if the intermediate data is media content data, it may be practical to consider 5GMS architectures, procedures and/or protocols for the streaming delivery of such intermediate media data.
3. Distributed/federated learning using multiple UE devices with local training sets, and a central server in the network. Typically a central model is distributed to UEs for local training. UEs use local data available to the device for local training, and training result updates are sent back to the central server, which aggregates and updates the central model. Global updates on the central model are then distributed to the UE devices for continuous training.

NOTE: Compression aspects will be addressed once the digital representation of AI/ML models will be identified together with their associated service requirements (eg. traffic flow characteristics, latency, bitrate…).

5.2.2 AI data components

AI related media data include:

* AI model data, including data describing the topology/structure of the AI model, data related to the data nodes of the model, i.e. tensors, and other data which may be dependent on the format used for the AI/Ml model.
* Intermediate data, defined as the output data from the inference process of an AI/Ml model that is not considered the final inference result (depending on the service and output layer of the split AI model, certain intermediate data may have media characteristics, or even be media data). Intermediate data is typically required to be delivered to a second device or entity, as the input to a subsequent second split inference.
* Inference output data, which is the data corresponding to the output result of the final AI inference process for the service. Depending on the nature of the AI data inferencing for the given AI data service, this inference output data may include: labels for identifying recognition like tasks from media, actual media data such as video and/or audio, or perhaps XR related data such as 3D models.

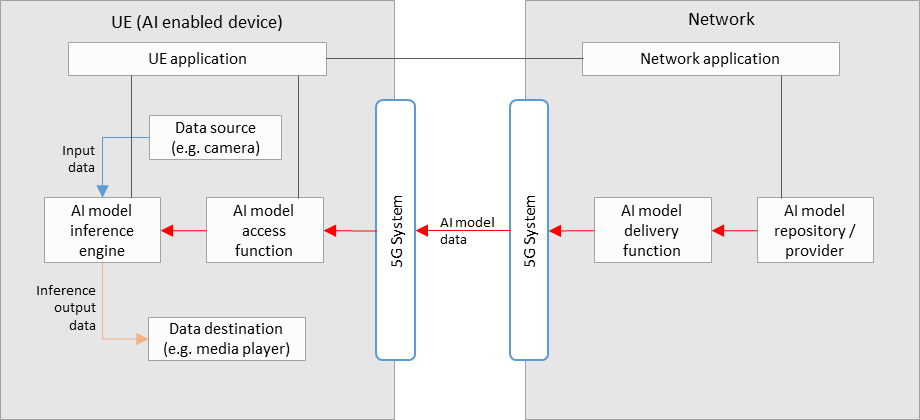
5.2.3 Media-related data logical functions

Media logical functions supporting the scenarios identified in the PD include:

* AI data delivery function
* AI data access function
* AI model inference engine

5.2.4 Complete/Basic AI/ML model distribution

5.2.4.1 Basic architectures



**Figure 5.2.4.1-1: Basic architecture for AI/ML model delivery with inference in the UE**

Figure 5.2.4.1-1 shows a simple basic architecture for AI/ML model delivery, as described in scenario 1) of clause 5.2, with an inference of a pre-trained AI/ML model in the UE, as described in scenario 2a) of clause 5.2.

In the network:

* An AI model in the repository is selected for the AI media service by the network application, and sent to the delivery function for delivery to the UE. Selection of an AI model could depend on UE and network characteristics, such as the memory and CPU capability/availability, as well as current network load and performance status.
* The AI model delivery function sends the AI model data to the UE via the 5GS. This delivery function may also contain functionalities related to QoS requests and monitoring, as well as those related to the optimization or compression of AI model data.

In the UE:

* A UE application provides an AI media service using the AI model inference engine and AI model access function.
* The AI model access function receives the AI model data via the 5G system, and sends it to the AI model inference engine. Receiver side optimization or decompression techniques for AI model data may be included.
* The AI model inference engine performs inference by using the input data from the data source (e.g. a camera, or other media source) as the input into the AI model received from the AI model access function. The inference output data is sent to the data destination (e.g. a media player).

Depending on the exact service scenario, AI model updates may be necessary during the service, and different AI model data delivery pipelines may be considered for such purposes. An AI model update may consist of a change in the AI model structure without disrupting the AI media service. If the AI model has requirements on UE memory, processing/computing capabilities or if running the AI model will increase the UE’s power consumption dramatically which will also influence the user experience of other services, it may actively request the update of the AI Model. For example, when the memory usage of the UE processing the AI Model exceeds a certain threshold, or if UE performance deteriorates, the UE can actively send a request to the network for an AI Model update. Alternatively, the network may also trigger the AI model update itself, where an interaction between the UE and network side might be needed to help the network collect current UE status information, e.g. Memory, CPU, current load, terminal location, current power consumption, current battery storage, etc.

5.2.4.2 Basic workflows

Figure 5.2.4.2-1 shows a basic workflow for AI/ML model delivery with inference in the UE. Steps for the procedures shown are described below.



**Figure 5.2.4.2-1: Basic workflow for AI/ML model delivery with inference in the UE**

During the initialization and establishment step, it is assumed that information related to the required features and detailed configurations are exchanged and negotiated between the network and UE. Information may include those related to UE device and network capabilities, AI/ML service information (e.g. service requirements, AI/ML model descriptions), and delivery methods. Such information may be used for the selection of a suitable AI/ML model for the service.

1. The *UE Application* and *Network Application* communicate to trigger AI model delivery, using the information from the initialization and establishment step.
2. An AI model is selected between the *UE Application* and *Network Application*.
3. The *Network Application* identifies the selected AI model in the *AI model Repository/Provider*.
4. The *AI Model Access Function* establishes an AI model delivery session with the *AI Model Delivery Function*.
5. The *AI Model Access Function* receives the AI model.
6. The *AI Model Access Function* passes the AI/ML model to the *AI model Inference Engine* in the UE.
7. The *Data Source* passes media data to the *AI model Inference Engine.*
8. The *AI Model Inference Engine* performs AI inferencing.
9. The *AI Model Inference Engine* passes the inference output result to the *UE Data Destination* for consumption.

Figure 5.2.4.2-2 shows a basic workflow for progressive model delivery. Steps for the procedures shown are described below.



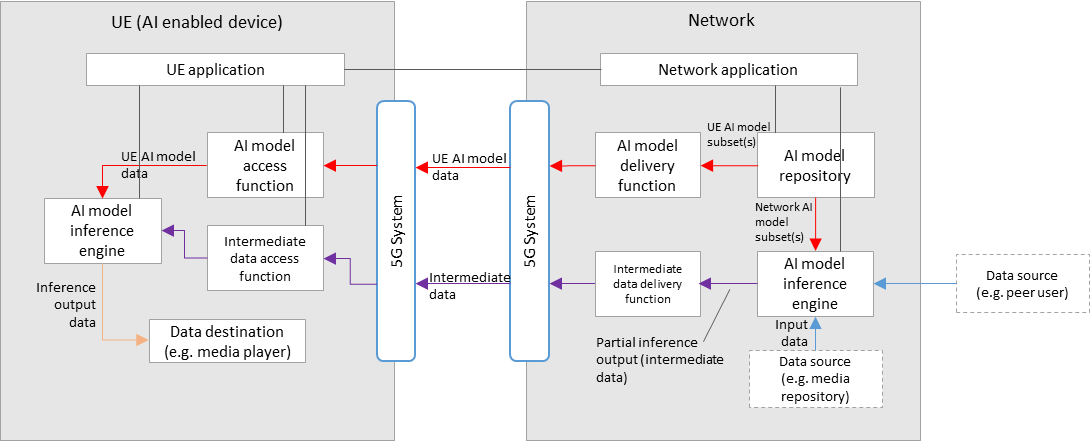
**Figure 5.2.4.2-2: Basic workflow for progressive model delivery**

Progressive model delivery refers to a model delivery paradigm wherein a low precision model is delivered to a UE first to expedite inference at the UE and to improve QoE. Subsequent model updates are delivered to the UE and the model at the UE is updated to a higher precision. The update may be applied to the model in different ways, depending on how the low precision model is obtained. For example, in case of bit-incremental model delivery the model update may be applied in an additive manner.

* 1. The UE Application and Network Application communicate to establish a progressive model delivery session. The UE Application may receive Service Access information to learn about available services and configurations, including available models, precisions and possible updates. This information may be in a 3GPP URI of/or model manifest file(s). The model manifest file contains size, complexity information etc. of the different versions.
  2. An AI model is selected by the UE Application, based on, e.g. model size and currently available network capacity.
  3. The UE application requests the selected model from the Network Application
  4. The Network Application identifies the selected AI model in the AI model Repository/Provider.
  5. The AI Model Access Function establishes an AI model delivery session with the AI Model Delivery Function.
  6. The AI Model Access Function receives the AI model of the precision requested by the UE.
  7. The AI Model Access Function passes the AI/ML model to the AI model Inference Engine in the UE.
  8. Inference loop: The Data Source passes data to the AI model Inference Engine, AI Model Inference Engine performs AI inferencing, and AI Model Inference Engine passes the inference output result to the UE Data Destination for consumption.
  9. The UE application triggers a model precision update (parallel to the inferencing loop of step 8). The update is a precision update of the model currently at the UE (steps 6-7) rather than a new model.
  10. The model update is delivered to the AI model access function
  11. The model in the inference engine is updated to a higher precision using the model update from 10.
  12. Steps 9-11 may be repeated as 12-13 depending upon number of precision levels and corresponding model updates

5.2.5 Split AI/ML operation

5.2.5.1 Basic architectures



**Figure 5.2.5.1-1: Basic architecture for split inference between the network and UE, with media data source in the network or from the UE via the network**

Figure 5.2.5.1-1 shows a simple basic architecture for split inferences between the network and the UE, as described in scenario 2b) of clause 5.2, where the media data source comes from the network, or peer user. The first part of the AI model is executed on the network side and the second part on the UE.

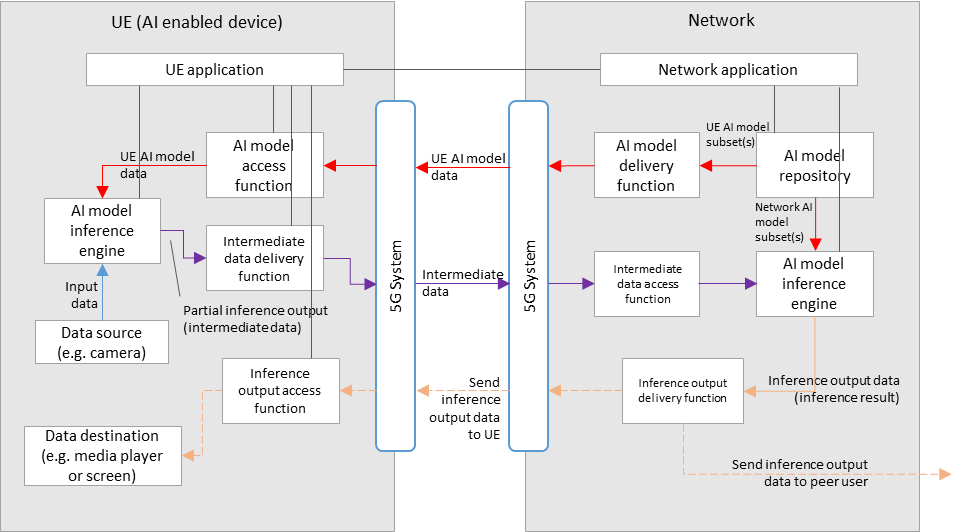
For the split inference (network-UE) scenario, additional components are required:

In the network:

* An AI model inference engine that receives both the network AI model subset(s), and input data, for network inference. The input data may come from the local media repository or the peer UE. An intermediate data delivery function receives the partial inference output (intermediate data) from the network inference engine, and sends it to the UE via the 5GS. This delivery function may also contain functionalities related to QoS requests and monitoring, as well as those related to the optimization or compression of intermediate data.

In the UE:

* An intermediate data access function receives the intermediate data from the network via the 5GS, and sends it to the UE inference engine for UE inference. If the intermediate data delivery function performs optimization or compression on intermediate data, this function may apply the corresponding reconstruction or decompression techniques.
* The final inference output data is sent to the data destination (e.g. a media player or screen).



**Figure 5.2.5.1-2: Basic architecture for split inference between the UE and network, with media data source in the UE**

Figure 5.2.5.1-2 shows a basic architecture for split inferences between the UE and the network, as described in scenario 2b) of clause 5.2, where the media data source originates from the UE, the first part of the inference is performed in the UE, the second part in the network. The resulting output data is finally sent back to the UE or the peer user.

For the split inference (UE - network) scenario, additional components are required:

In the UE:

* An AI model inference engine that receives both the network AI model subset(s), and input data (from a UE data source), for UE inference.
* An intermediate data delivery function receives the partial inference output (intermediate data) from the UE inference engine, and sends it to the network via the 5GS. This delivery function may also contain functionalities related to QoS requests and monitoring. If the intermediate data delivery function performs optimization or compression on intermediate data, this function may apply corresponding optimization or decompression techniques.
* An inference output access function receives the inference output data from the network via the 5GS, and sends it to the relevant data destination according to the AI media service.

In the network:

* An intermediate data access function receives the intermediate data from the UE via the 5GS, and sends it to network inference engine for network inference. If the intermediate data delivery function applies optimization or compression on intermediate data, this function may apply corresponding optimization or decompression techniques.

The final inference output data is sent to the UE or the peer user via the 5GS, through the inference output delivery function.

For both split inference scenarios, extra factors should be considered, including those such as:

* Configuration of the split inference between the network and UE. (e.g. definition and selection of the AI/ML model composition into “UE AI model subset” and “network AI model subset”)
* Resource allocation and management for network inference, including ingestion of network AI model data and media data
* Intermediate data delivery pipelines between the network and UE, in particular considering the use of 5GMS or RTC defined pipelines to stream intermediate data that is media content data.
* The functionalities of certain components in figure 5.2.1-1 and figure 5.2.2-1 may overlap, and depending on the use case a combined architecture may also be considered FFS.
* Certain components may also overlap with functions defined in 5GMS or RTC architectures, clarifications FFS.

5.2.5.2 Basic workflows

Figure 5.2.5.2-1 shows a basic workflow for split inference between the network and UE. Steps for the procedures shown are described below.



**Figure 5.2.5.2-1: Basic workflow for split inference between the network and UE**

0. The session is established between the UE and the network.

**AI Split Inference Negotiation** (This step may be performed at the beginning or during the session when the UE or network status has changed):

1. The *UE Application* gets the UE’s capability information, which may include the AI inference processing capabilities, supported AI framework information, connection capabilities, etc.

**Alternative Case#1: Network decides the split inference:**

2a. When the *UE Application* discovers the UE’s local capabilities can’t meet the AI service requirement, it sends an AI split inference request to the *Network Application* with UE’s capability information and the service requirement information.

3a. The *Network Application* gets the network’s capability information, which includes the AI inference processing capabilities, supported AI framework information from the *AI Inference Engine*.

4a. The *Network Application* selects a proper AI model (including the UE AI model subset and the network AI model subset) for split inference from all matched AI models (with different candidate split points) based on the service requirement information, the UE’s capability information and the network’s capability information.

5a. The *Network Application* sends an AI Inference Resource Allocation request to the *AI Model Inference Engine* with the selected network AI model subset information (including the split point and the intermediate data information).

6a. The *AI Model Inference Engine* responds with a successful result to the *Network Application*.

7a. The *Network Application* sends the AI Split Inference Response with the selected UE AI model subset information (including the split point and the intermediate data information) to the *UE Application*.

**Alternative Case#2: UE decides the split inference:**

2b. The *UE Application* sends an AI Model Information Request to the network with the UE’s capability information and the service requirement information*.*

3b. The *Network Application* collects all matched AI models with different candidate split points based on the service requirement information, the UE’s capability information and the network’s capability information.

4b. The *Network Application* sends the AI Model Information Response with all matched UE AI model subset(s) information (including the split point and the intermediate data information) to the *UE Application*.

5b. The *UE Application* selects a proper AI model based on the UE’s capability information and the received information in the AI Model Information Response.

6b. The *UE Application* sends an AI Split Inference Request to the *Network Application* with the selected AI model information.

7b. The *Network Application* sends an AI Inference Resource Allocation request to the *AI Model Inference Engine* with the network model subset information corresponding to the AI model selected by the *UE Application*.

8b. The *AI Model Inference Engine* responds with a successful result to the *Network Application*.

9b. The *Network Application* sends the AI Split Inference Response to the *UE Application*.

**AI Model Subset Delivery:**

10. The *Network Application* identifies the selected UE and network AI model subsets in the *AI model Repository*.

11. The *AI Model Inference Engine* in the network receives the network AI model subset.

12. The *AI Model Access Function* establishes a UE AI model subset delivery session with the *AI Model Delivery Function*.

13. The *AI Model Access Function* receives the UE AI model subset.

14. In the UE, the *AI Model Access Function* passes the UE AI model subset to the *AI model Inference Engine*.

**AI split inference:**

**Alternative case#1: data source in the network**

15a. The network *AI model Inference Engine* receives media data from the network *Data Source* or a peer user.

16a. The network *AI model Inference Engine* performs network AI inferencing.

17a. The *Intermediate Data Access Function* establishes an intermediate data delivery session with the *Intermediate Data Delivery Function*.

18a. In the UE, the *Intermediate Data Access Function* receives intermediate data and passes it to the *AI Model Inference Engine*.

19a. The *AI Model Inference Engine* in the UE performs AI inferencing.

20a. The *AI Model Inference Engine* passes the inference output result to the *UE Data Destination* for consumption.

**Alternative case#2: data source in the UE**

15b. In the UE, the *Data Source* passes media data to the *AI model Inference Engine*.

16b. The UE *AI model Inference Engine* performs UE AI inferencing.

17b. The *Intermediate Data Access Function* establishes an intermediate data delivery session with the *Intermediate Data Delivery Function*.

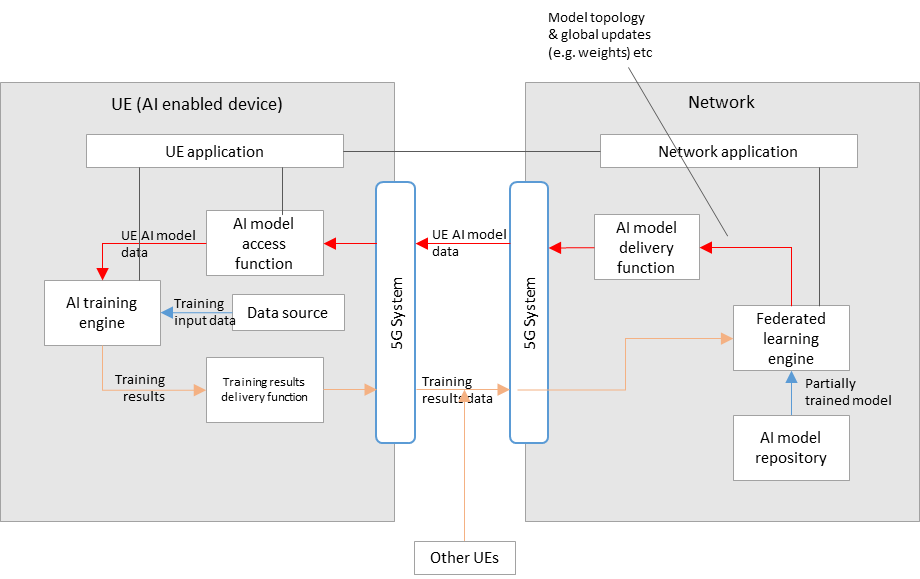
18b. In the network, the *Intermediate Data Access Function* receives intermediate data and passes it to the *AI Model Inference Engine*.

19b. In the network, the *AI Model Inference Engine* performs network AI inferencing.

20b. The network *AI Model Inference Engine* sends the inference output result to the *UE Data Destination* or a peer user.

5.2.6 Distributed/federated learning

5.2.6.1 Basic architectures



**Figure 5.2.6.1-1: Basic architecture for distributed/federated learning between the network and multiple UEs**

Figure 5.2.6.1-1 shows a simple basic architecture for distributed/federated learning between the network and UE(s), as described in scenario 3) of clause 5.2.

In the network:

* A federated learning engine receives a partially trained model from the AI model repository, that is passed to the AI model delivery function for delivery to multiple UEs via the 5GS.
* Training results data from multiple UEs is also received by the federated learning engine via the 5GS, which is then aggregated for the continuous training of the global model.
* Updates to the global model (e.g. in terms of topology or weights) are delivered to the UEs during the learning process.

In the UE(s):

* AI model data is received by an AI model access function via the 5GS, which then passes the data to the AI training engine.
* An AI training engine in the UE trains the AI model using local device data as the training input.
* Training results (e.g. in the form of updated weights) are delivered to the network via the training results delivery function.

5.2.6.2 Basic workflow

Figure 5.2.5.2-1 shows a basic workflow for distributed/federated learning with training in the UE, the results of which are aggregated in the network. Steps for the procedures shown are described below.



**Figure 5.2.6.2-1: Basic workflow for distributed/federated learning between a UE and the network**

During the initialization and establishment step, it is assumed that information related to the required features and detailed configurations are exchanged and negotiated between the network and UE. Information may include those related to UE device and network capabilities, AI/ML service information (e.g. service requirements, AI/ML model descriptions), and delivery methods. Such information may be used for the selection of a suitable partially trained AI/ML model for the service.

1. The *UE Application* and *Network Application* communicate to trigger distributed/federated learning, using the information from the initialization and establishment step.
2. A partially trained AI model is selected between the *UE Application* and *Network Application*.
3. The *Network Application* identifies the selected partially trained AI model in the *AI model Repository/Provider*.
4. The *Federated Learning Engine* optionally announces the eligibility criteria for participating in the federated evaluation/learning to the device. The criteria could contain various information such as the device's operating system, processor speed, available memory, characteristics of the data library, geographical location of the device, language setting, and other attributes.
5. The *AI Model Access Function* of an eligible device receives the partially trained AI model or its updated version
6. The *Federated Learning Engine* optionally announces the failure reporting criteria for the participating devices.

Option A: Model evaluation:

1. The *Federated Learning Engine* requests the UE to start the model evaluation. The evaluation mechanism and criteria are defined by the *Federated learning Engine.*

Note:Whether a user wants its device to participate in the evaluation, depends on the business agreement between the user and the network.

1. The *Data Source* passes the training input data to the *AI model Training Engine.*
2. The *AI Model Training Engine* performs the evaluation.
3. The evaluation results (or the failure messages, in the case of a failure) are delivered to the *Federated Learning Engine*.
4. Optionally, the device eligibility criteria may get updated depending on the evaluation results.

Option B: Federated training:

1. The *Federated Learning Engine* requests the UE to start the training.

Note:Whether a user wants its device to participate in the training, depends on the business agreement between the user and the network.

1. The *Data Source* passes the training input data to the *AI model Training Engine.*
2. The *AI Model Training Engine* performs the retraining of the model.
3. The updated model (or the failure messages, in the case of a failure) is delivered to the *Federated Learning Engine*.
4. The *Federated Learning Engine* performs training aggregation of training results from multiple UEs and updates the partially trained AI model.
5. The updated partially trained AI model is delivered to the UE as from step 5.

Note: As shown in the above call flow, the model evaluation and the federated learning may also occur in a sequence.

5.3 Architecture and procedures for AI data delivery over 5G

5.3.1 Architecture and components

5.3.1.1 Introduction

Une image contenant texte, capture d’écran, diagramme, Plan

Description générée automatiquement

**Figure 5.3.1.1-1 AI data delivery general architecture**

A generalized 5G Media Delivery architecture supporting AI media functionality is shown in figure 5.3.1.1-1. Depending on the service scenario and/or use case, certain dedicated AI/ML logical subfunctions may be mapped to, or instantiated by the generalized media architecture functions..

5.3.1.2 Network functions and UE entities

In addition to the media related definitions described in TS 26.501, additional definitions for AI data related functions include:

- **Media Client** running on the UEcontains two subfunctions:

- **Media Session Handler:** A function on the UE that communicates with the network side Media AF to establish and control the configuration of an AI data session. The function may include:

- Features that monitors, shares and/or reports UE capabilities with/to the Media AF. This may be used for the selection of the model for a UE inference or for the selection of the UE model subset part for a split inference topology between the UE and the network.

- **Media Access Function:** A function on the UE that communicates with the Media AS and the Media Session Handler to establish an AI data delivery session. The function contains:

- An AI Inference Engine, which has the capability to perform the inferencing of received (split) AI models.

- An AI Data Access/Delivery function, which handles the access and delivery of user plane AI/ML data, as well as conventional media data including

- Download the AI model data for inference process. This includes instantiating an AI data access client to access and retrieve AI models or AI model subsets from local files or over the network (e.g., by streaming or downloading the model from a remote server). The inference engine may comprise format decapsulation and model decoding functions as well as a runtime engine that executes the model from the memory.

- Access/deliver intermediate data when an inference is split between the UE and the network.

- Encode data to deliver with serialization and/or compression techniques or conversely decode the received data with deserialization or decompression technique.

- **Media-aware Application:** An external function controlled by the external media application provider implementing the AI/ML application logic, which includes triggering the delivery of an AI model to the inference engine and obtaining inference results from the inference engine.

- **Media AS (Application Server):** An Application Server that hosts 5G AI data functions. It includes

*-* An AI data Access/Delivery function, which handles the access and delivery of user plane AI/ML data, as well as conventional media data as described above.

*-* An AI Inference Engine, which has the capability to perform the inferencing of (split) AI models.

- **Media AF (Application Function):** An Application Function that provides various control and configuration functions to the Media Session Handler on the UE and/or to the Media Application Provider. It may relay or initiate a request for different Policy or Charging Function (PCF) treatment or interact with other network functions via the NEF (Network Exposure Function). The Application function can include for example:

- Supporting features such as monitoring, sharing and/or reporting Network capabilities with/to the Media Session Handler*.* This may be used for the selection of the model for a UE inference or for the selection of the UE model subset part for a split inference topology between the UE and the network via the Media Access Function.

5.3.2 Procedures for Split AI/ML operation

Figure 5.3.2-1 shows an example procedure for split AI/ML operation, including three main parts:

* AI split inference management, and
* AI data delivery session
* Split inference processing

Une image contenant texte, diagramme, reçu, Parallèle

Description générée automatiquement

Figure 5.3.2-1: Procedures for split AI/ML operation

1. Service provisioning and announcement of AI data service on the network side, in particular between the Media AF (application function) and the 5GAI application provider.
2. Service access information acquisition. During this step, the available or required AI model(s) for the service can be made known to the UE, by means of information made available via a URL link pointing to a file or manifest which may list such available AI models. Such additional information may contain a list of features available from each AI model, including its variants, including specific information such as the inferencing accuracy, the size, , the amount of nodes, structure, complexity and latency requirements of the AI model.

AI split inference management:

1. Discovering AI data inferencing capabilities and functions in both the UE and network. In this step, the Media Session Handler in the UE and the Media AF in the network may use its capabilities to calculate the range of inference latencies for the AI model to be used for the split AI/ML inference service.
2. Requesting AI split inference. Either the UE or the network requests the other side for an AI split inference service. If information describing the AI model was not made known via the service access information in step 2, then such information may also be shared during this step.
3. Negotiate splitting the AI inference process. A split point is negotiated between the UE and the network, using information from steps 2, 3 and 4, in order to satisfy the service, capability and AI model inference latency requirements. The decision of whether the split point is static or whether it can be updated dynamically during the service may be negotiated. Related metadata may be shared between the network and UE depending on the configuration and a set of split points can be negotiated.
4. Acknowledge the split and provide the AI data split inferencing access info. In this step, the network (Media AF) and UE (AI data session handler) both acknowledge the decided split point, and access information for the AI data is provided to the UE.
5. The split configuration outcome is notified to the Media-aware application.

Split AI data session

1. Request the start of intermediate data delivery. On confirmation, the application triggers the Media Client to request the start of AI data delivery using the AI intermediate data access information provided in step 7.
2. The Media Client request the intermediate data to be deliver from the Media AS.
3. Pipelines for the delivery of AI model data from the Media AS to the Media Client are setup, and suitable delivery sessions are established and initiated. Delivery may be in the manner of streaming delivery, or download delivery (such as that defined in TS 26.501, or any other form of delivery mechanism required by the AI data service.
4. Start inference process in the UE. In this step, the Media Client triggers the inference process (the AI inference engine function), namely the UE side of the split inferencing as decided by the result of step 5.
5. Start inference process in the server. In this step, the Media AF triggers the inference process in the Media AS (the AI inference engine function), namely the network side of the split inferencing as decided by the result of step 5.
6. Pipelines for the delivery of intermediate data from the Media AS to the Media Client are setup, and suitable delivery sessions are established and initiated. Delivery may be in the manner of streaming delivery, such as that defined in TS 26.501, or any other form of delivery mechanism required by the AI data service.

Split inference processing

1. The split inference runs between the UE and the network. Depending on the specific split inference scenario, the UE and the network may deliver and/or access Intermediate data, Inference output data and/or metadata using the pipelines defined in the AI data delivery session.

Split point update and inference processing

1. A split point update is triggered, for example from the media aware application to adapt to the new conditions (e.g. UE capabilities or network capacity has changed). The new split point metadata information is either negotiated between the UE and the network or pass alongside the delivery pipeline from the UE to the network side.

Session reporting and update

1. The Media Session Handler may collect and send status reports regarding the UE’s AI media service status (for example AI inference status, latency, resource status, capability status, dynamic media properties etc.) to the Media AF.
2. The Media AS may send status reports regarding the network’s AI media service status to the Media AF.
3. The Media Session Handler may receive network status, or network AI status reports from the Media AF, as collected in step 16.
4. The Media Session Handler may receive media status reports either from the network or internally from the UE.
5. Depending on the configurations negotiated in step 5, as well as related information from the status reports in steps 16, 17 and 18, updates of the AI model selection, split point configuration or the AI data delivery pipelines for the session may take place between the UE and network.

### 5.3.3 Procedure for AI/ML model distribution and operation

Figure 5.3.3-1 shows a procedure for AI/ML model distribution and operation.

Similar to the procedures for 5GMS downlink Media Streaming, assuming that the network operator provides such an AI/ML model distribution service, as well as the availability of AI models from the Media Application Provider, the procedure consists of an ingest session (where AI models are uploaded to the Media AS), and a provisioning session, during which the Media Client can access the AI models and the Media Application Provider can control and monitor the AI models and its delivery.

Une image contenant texte, diagramme, nombre, Police

Description générée automatiquement

Figure 5.3.3-1: Procedure for AI/ML model distribution and operation

Steps 1 to 8 assume the same steps as defined in TS 26.501 for downlink media streaming, but for AI/ML distribution; the Service Announcement Information (whether acquired in whole in step 4 or through a reference and later in whole in step 6) contains information allowing the Media Client to activate the reception of one or more AI models. In step 9, the Media Client performs AI inferencing using media as an input into the AI model delivered and received in step 8.

Depending on the distribution use case, media delivery features (such as dynamic policy, network assistance, metrics reporting etc.) may also be applicable to AI/ML distribution.

5.4 Architecture and procedures for AI data delivery over IMS

5.4.1 Architecture and components



**Figure 5.4.1-1 AI data delivery over IMS architecture**

Figure 5.4.1-1 shows a mapping of AI data delivery functions to the IMS data channel architecture which was defined in clause AC.2 of TS 23.228. The AI data delivery over IMS includes the following functionalities:

UE:

- AI Inference Engine: It has the capability to perform the inferencing of received (split) AI models.

- AI Data Access/Delivery: It handles the access and delivery of AI data including

- Download the AI model data for inference process. This includes instantiating an AI data access client to access and retrieve AI models or AI model subsets over the network (e.g., by streaming or downloading the model from a remote server).

- Access/deliver intermediate data when an inference is split between the UE and the network.

- Encode data to deliver with serialization and optionally compression techniques. Or conversely decode the received data with deserialization or optionally decompression techniques.

MF:

- AI Data Access/Delivery: It handles the access and delivery of AI data as described above.

- AI Inference Engine: It has the capability to perform the inferencing of (split) AI models.

DC AS:

- It monitors, shares and/or reports network capabilities with/to the UE. This may be used for the selection of the model for a UE inference or for the selection of the UE model subset part for a split inference topology between the UE and the network.

AI Model Repository (AIMR):

- It provides the AI models (including subsets) storage and downloading function. AIMR can be inside the PLMN, e.g. a new network function, or outside the PLMN, e.g. a webserver of the 3rd party provider.

5.4.2 Procedures for Split AI/ML operation



**Figure 5.4.2-1 Procedures for split AI/ML operation**

0. The audio/video and data channel sessions are established between the UE-A and the UE-B.

**AI Split Inference Negotiation** (This step may be performed at the beginning or during the session when the UE or network status has changed):

**Alternative Case#1: Network decides the split inference:**

1a. The UE-A gets the UE’s capability information, which may include the AI inference processing capabilities, supported AI framework information, connection capabilities, etc. When the UE-A discovers the UE’s local capabilities can’t meet the AI service requirement, it decides to trigger the split inference process.

2a. The UE-A sends an AI split inference request to the DC AS with UE’s capability information and the service requirement information.

3a. The DC AS gets the MF’s capability information, which includes the AI inference processing capabilities, supported AI framework information from the AI Inference Engine, and selects a proper AI model (including the UE AI model subset and the network AI model subset) for split inference from all matched AI models (with different candidate split points) based on the service requirement information, the UE’s capability information and the network’s capability information.

4a. The DC AS sends an AI Inference Resource Allocation request to the DCSF, the request includes the selected network AI model subset information (including the split point and the intermediate data information). The request is transferred to the IMS AS.

5a. The IMS AS instructs a media resource allocation request to the MF.

6a. The MF responds with a successful result to the IMS AS, the IMS AS transfers the response to the DCSF, and the DCSF transfers it to the DC AS.

7a. The DC AS sends the AI Split Inference Response with the selected UE AI model subset information (including the split point and the intermediate data information) to the UE-A.

**Alternative Case#2: UE decides the split inference:**

1b. The UE-A gets the UE’s capability information, which may include the AI inference processing capabilities, supported AI framework information, connection capabilities, etc. When the UE-A discovers the UE’s local capabilities can’t meet the AI service requirement, it decides to get the AI models from the network for split inference process.

2b. The UE-A sends an AI Model Information Request to the DC AS with the UE’s capability information and the service requirement information.

3b. The DC AS collects all matched AI models with different candidate split points based on the service requirement information, the UE’s capability information and the network’s capability information.

4b. The DC AS sends the AI Model Information Response with all matched UE AI model subset(s) information (including the split point and the intermediate data information) to the UE-A.

5b. The UE-A selects a proper AI model based on the UE’s capability information and the received information in the AI Model Information Response.

6b. The UE-A sends an AI Split Inference Request to the DC AS with the selected network AI model information.

7b. The DC AS sends an AI Inference Resource Allocation request to the DCSF, the request includes the selected network AI model subset information (including the split point and the intermediate data information). The request is transferred to the IMS AS.

8b. The IMS AS instructs a media resource allocation request to the MF.

9b. The MF responds with a successful result to the IMS AS, the IMS AS transfers the response to the DCSF, and the DCSF transfers it to the DC AS.

10b. The DC AS sends the AI Split Inference Response to the UE-A.

**AI Model Subset Delivery:**

11. The MF downloads the network AI model subset from the AIMR.

12. The UE-A downloads UE AI model subset from the AIMR over the established data channel.

**AI split inference:**

**Alternative case#1: data source from peer user**

13a. The MF receives media data from the peer user.

14a. The MF performs network AI inferencing.

15a. The MF deliveries the intermediate data to the UE-A.

16a. The UE-A performs AI inferencing and output the inference result.

**Alternative case#2: data source from local user**

13b. The UE-A performs UE AI inferencing based on the media data generated locally.

14b. The UE-A deliveries the intermediate data and passes it to the MF.

15b. The MF performs network AI inferencing.

16b. The MF sends the inference output result to the UE-B.

\* \* \* End of Changes \* \* \* \*