Internet Research Task Force
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

Y-G. Hong
Daejeon Univ.
J-S. Youn
DONG-EUI Univ.
S-W. Hong
ETRI
P. Martinez-Julia
NICT
Q. Wu
Huawei

20 October 2025

Considerations of network/system for AI services
draft-irtf-nmrg-ai-deploy-02

Abstract

As the development of AI technology has matured and AI technology has begun to be applied in various fields, it has changed from running only on very high-performance servers to running on commodity servers, with affordable, small-scale hardware, including microcontrollers, low-performance CPUs, and AI chipsets. This document outlines how to configure the network and system for an AI inference service, providing AI services in a distributed manner. It also outlines the factors to consider when a client connects to a cloud server and an edge device to requests an AI service. It describes some use cases for deploying network-based AI services, such as self-driving vehicles and network digital twins.

Table of Contents

1. Introduction

In the Internet of Things (IoT), the amount of data generated from IoT devices has exploded along with the number of IoT devices due to industrial digitization and the development and dissemination of new devices. Various methods are being tried to effectively process the explosively increasing IoT devices and data of IoT devices. One of them is to provide IoT services in a place located close to IoT devices and users, away from cloud computing that transmits all data generated from IoT devices to a cloud server [RFC9556].

IoT services also started to break away from the traditional method of analyzing IoT data collected so far in the cloud and delivering the analyzed results back to IoT objects or devices. In other words, AIoT (Artificial Intelligence of Things) technology, a combination of IoT technology and artificial intelligence (AI) technology, started to be discussed at international standardization organizations such as ITU-T. AIoT technology, discussed by the ITU-T CG-AIoT group, is defined as a technology that combines AI technology and IoT infrastructure to achieve more efficient IoT operations, improve human-machine interaction, and improve data management and analysis [CG-AIoT].

The first work started by the IETF to apply IoT technology to the Internet was to research a lightweight protocol stack instead of the existing TCP/IP protocol stack so that various types of IoT devices, not traditional Internet terminals, could access the Internet [RFC6574][RFC7452]. These technologies have been developed by 6LoWPAN working group, 6lo working group, 6tisch working group, core working group, t2trg group, etc. As the development of AI technology matured and AI technology began to be applied in various fields, just as IoT technology was mounted on resource-constrained devices and connected to the Internet, AI technology is also changed from running only on very high-performance servers. The technology is being developed to run on small hardware, including microcontrollers, low-performance CPUs and AI chipsets. This technology development direction is called On-device AI or TinyML[tinyML].

In this document, we consider how to configure the network and system in terms of AI inference service to provide AI service in the IoT environment. In the IoT environment, the technology of collecting sensing data from various sensors and delivering it to the cloud has already been studied by many standardization organizations including the IETF and many standards have been developed. Now, after creating an AI model to provide AI services based on the collected data, how to configure this AI model as a system has become the main research goal. Until now, it has been common to develop AI services that collect data and perform inferences from the trained servers, but in

terms of the spread of AI services, it is not appropriate to use expensive servers to provide AI services. In addition, since the server that collects and trains data mainly exists in the form of a cloud server, there are also many problems in proceeding in the form of requesting AI service by connecting a large number of terminals to these cloud servers to provide AI services. Therefore, when an AI service is requested to an edge device located at a close distance, it may have effects such as real-time service support, network traffic reduction, and important data security rather than requesting an AI service to an AI server located in a distant cloud[RFC9556].

Even if an edge device is used to serve AI services, it is still important to connect to an AI server in the cloud for tasks that take a lot of time or require a lot of data. Therefore, an offloading technique for properly distributing the workload between the cloud server and the edge device is also a field that is being actively studied.

Furthermore, beyond the deployment of basic AI inference on distributed devices (On-device lightweight AI or TinyML), the concept of Agentic AI is rapidly emerging. Agentic AI refers to an AI system capable of autonomous goal setting, complex task decomposition, planning, execution, and continuous learning and reflection, rather than simple one-off inference responses. Deploying such Agentic AI systems in a distributed environment (Cloud-Edge-Local) is essential for realizing truly autonomous and intelligent network management and service provision. This document’s considerations for distributed AI deployment are directly applicable to the foundational network infrastructure required for Agentic AI.

In this contribution, in the following proposed network structure, the points to be considered in the environment where a client connects to a server and an edge device and requests an AI service are derived and described. That is, the following considerations and options could be derived.

* AI inference service execution entity

* Hardware specifications of the machine to perform AI inference services

* Selection of AI models to perform AI inference services

* A method of providing AI services from cloud servers or edge devices

* Communication method to transmit data to request AI inference service

The proposed considerations and items could be used to describe the use case of self-driving vehicles and network digital twins. Since providing AI services in a distributed method can provide various advantages, it is desirable to apply it to self-driving vehicles and network digital twins.

2. Procedure to provide AI services

Since research on AI services has been started for a long time, there may be shapes to provide various types of AI services. However, due to the nature of AI technology, in general, a system for providing AI services consists of the following steps [AI_inference_archtecture] [Google_cloud_iot].

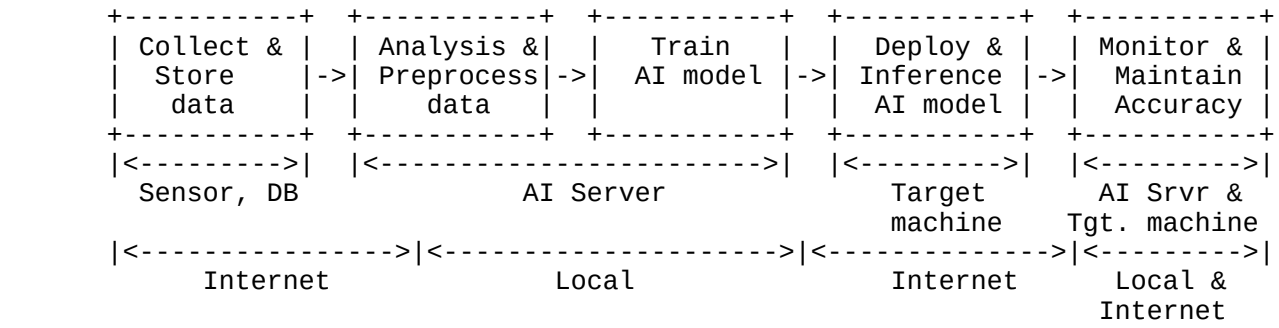

Figure 1: AI service workflow

* Data collection & Store
* Data Analysis & Preprocess
* AI Model Training
* AI Model Deploy & Inference
* Monitor & Maintain Accuracy

In the data collection step, data required for training is prepared by collecting data from sensors and IoT devices or by using data stored in a database. Equipment involved in this step includes sensors, IoT devices and servers that store them, and database servers. Since the operations performed at this step are conducted through the Internet, many IoT technologies studied by the IETF so far have developed technologies suitable for this step.

In the data analysis and pre-processing step, the features of the prepared data are analyzed and pre-processing for training is performed. Equipment involved in this step includes a high-performance server equipped with a GPU and a database server, and is mainly performed in a local network.

In the model training step, a training model is created by applying an algorithm suitable for the characteristics of the data and the problem to be solved. Equipment involved in this step includes a high-performance server equipped with a GPU, and is mainly performed on a local network.

In the model deploying and inference service provision step, the problem to be solved (e.g., classification, regression problem) is solved using AI technology. Equipment involved in this step may include a target machine, a client, a cloud, etc. that provide AI services, and since various equipment is involved in this stage, it is conducted through the Internet. This document summarizes the factors to be considered at this step.

In the accuracy monitoring step, if the performance deteriorates due to new data, a new model is created through re-training, and the AI service quality is maintained by using the newly created model. This step is the same as described in the model training, model deploying, and inference service provision steps described in the previous step because re-training and model deploying are performed again.

3. Network configuration structure to provide AI services

In general, after training a AI model, the AI model can be built on a local machine for AI model deploying and inference services to provide AI services. Alternatively, we can place AI models on cloud servers or edge devices and make AI service requests remotely. In addition, for overall service performance, some AI service requests to the cloud server and some AI service requests to edge devices can be performed through appropriate load balancing.

3.1. AI inference service on Local machine

The following figure shows a case where a client module requesting AI service on the same local machine requests AI service from an AI server module on the same machine.

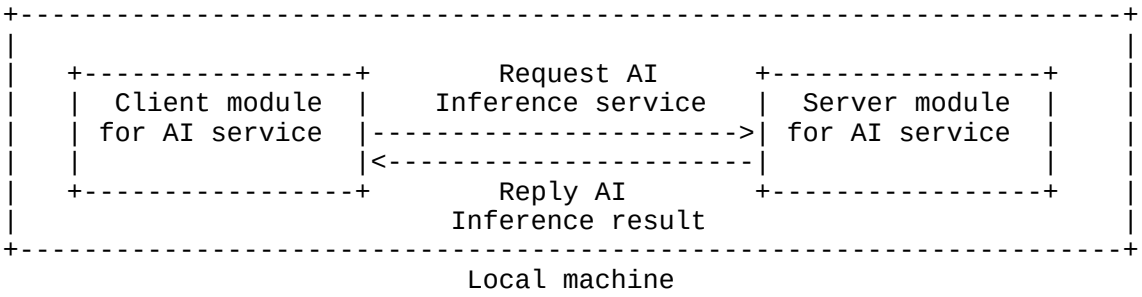

Figure 2: AI inference service on Local machine

This method is often used when configuring a system focused on training AI models to improve the inference accuracy and performance of AI models without considering AI services or AI model deploying and inference in particular. In this case, since the client module that requests the AI inference service and the AI server module that directly performs the AI inference service are on the same machine, it is not necessary to consider the communication/network environment or service provision method too much. Alternatively, this method can be used when we want to simply decorate the AI inference service on one machine without changing the AI service in the future, such as an embedded machine or a customized machine.

In this case, a high level of hardware performance is not required to train the AI model, but hardware performance sufficient to run the AI inference service is required, so it is possible on a machine with a certain amount of hardware performance.

3.2. AI inference service on Cloud server

The following figure shows the case where the client module that requests AI service and the AI server module that directly performs AI service run on different machines.

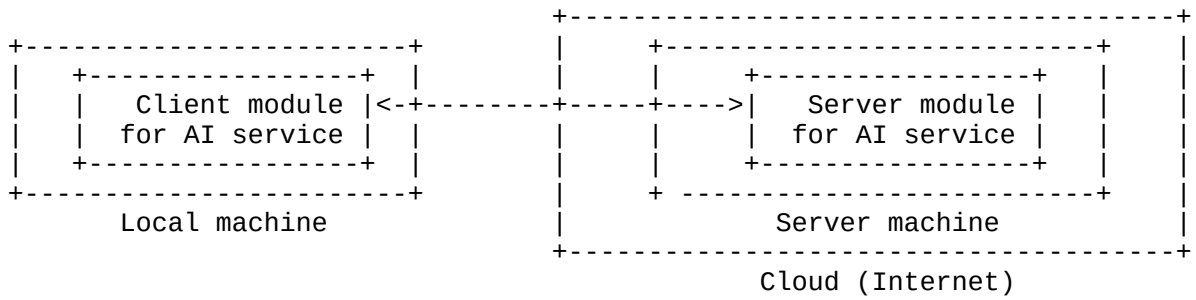

Figure 3: AI inference service on Cloud server

In this case, the client module requesting the AI inference service runs on the local machine, and the AI server module that directly performs the AI inference service runs on a separate server machine, and this server machine is in the cloud network. In this case, the performance of the local machine does not need to be high because the local machine simply needs to request the AI inference service and, if necessary, deliver only the data required for the AI service request. For the AI server module that directly performs AI inference service, we can set up our own AI server, or we can use commercial clouds such as Amazon, Microsoft, and Google.

3.3. AI inference service on Edge device

The following figure shows the case where the client module that requests the AI service and the AI server module that directly performs the AI service are separated, and the AI server module is located on the edge device. As more and more data is generated at the edge, it is necessary to perform AI processing at the edge due to performance, privacy, and cost of moving data to a central location, for example, some enterprises that own both local machine and edge network will not feel comfortable uploading their confidential datasets to the central cloud, instead these enterprises retrieve AI model from the cloud and use such AI model for AI inference locally at the edge device.

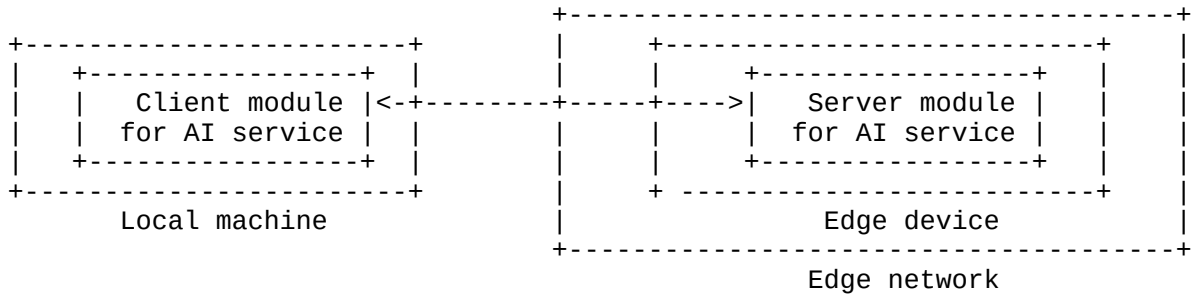

Figure 4: AI inference service on Edge device

Even in this case, the client module that requests the AI inference service runs on the local machine, the AI server module that directly performs the AI inference service runs on the edge device, and the edge device is in the edge network. Even in this case, the client module that requests the AI inference service runs on the local machine, the AI server module that directly performs the AI inference service runs on the edge device, and the edge device is in the edge network. The AI module that directly performs the AI inference service on the edge device can directly configure the edge device or use a commercial edge computing module.

The difference from the above case where the AI server module is in the cloud is that the edge device is usually close to the client, whereas the performance is lower than that of the server in the cloud, so there are advantages in data transfer time and inference time, but in unit time Inference service performance is poor.

3.4. AI inference service on Cloud server and Edge device

The following figure shows the case where AI server modules that directly perform AI services are distributed in the cloud and edge devices.

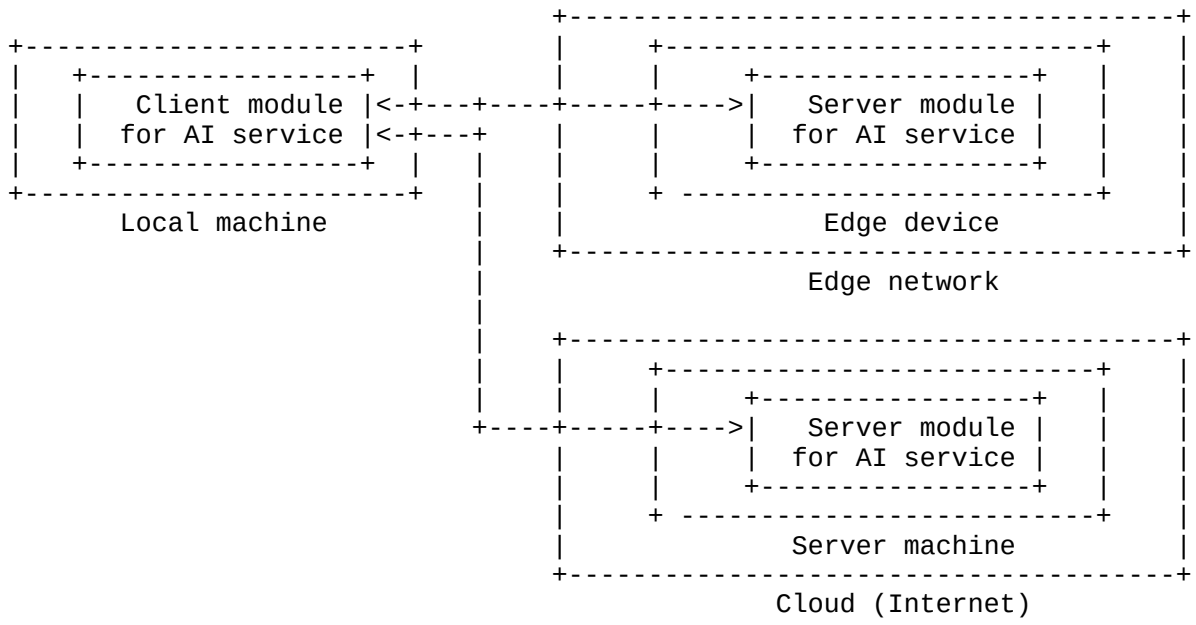

Figure 5: AI inference service on Cloud sever and Edge device

There is a difference between the AI server module performed in the cloud and the AI server module performed on the edge device in terms of AI inference service performance. Therefore, the client requesting the AI inference service may request by distributing the AI inference service request to the cloud and edge device appropriately in order to perform the desired AI service. In other words, in the case of an AI service with low inference accuracy but short inference time, we can request an AI inference service to the edge device.

The distribution of AI inference service requests between the cloud and edge devices is a critical decision. In the context of Agentic AI, the AI client module itself evolves into an AI Agent Planning Module. This module autonomously determines the optimal execution point (Local, Edge, or Cloud) based on a dynamic assessment of inference accuracy requirements, latency constraints (e.g., in-time/ on-time delay sensitivity), and resource costs. This autonomous decision-making process is a core characteristic of Agentic AI.

3.5. AI inference service on horizontal multiple servers

In the previous section, to provide AI inference service, the network configuration that consisted of local machines, edge devices, and cloud servers is a kind of vertical hierarchy. Because the capabilities of each machine are different, the overall performance of the network using vertical hierarchy is dependent of each machine. Generally, a cloud server has a most powerful performance and then an edge device has the second powerful performance.

In this network configuration, AI service may have different performance according to the load level of the server, computing capability of the server machine and link-state between the local machine and the server machines of the horizontal level. Thus, to look for the server machine that can support the best AI service, it is necessary for the network element that can monitor network link-state and current state of the computing capability of the server machines and the network load-balance that can perform a scheduling policy of load balancing. Therefore, when a client requests an AI inference service, those requests are shared between edge devices and cloud servers, and routed to a single cloud server machine in the network based on the load balancer’s decision. Alternatively, these requests are processed by all the Cloud Server machines that contribute to the final inference result.

When utilizing horizontal multiple servers, the integration of Multi-Agent Systems (MAS) based on Agentic AI becomes crucial. Instead of relying solely on a centralized load balancer, multiple AI agents, each deployed on a separate server machine (or edge device), can autonomously negotiate tasks and collaborate toward a common objective. This multi-agent collaboration structure allows for the decomposition of a complex AI task into sub-tasks, with each agent specializing in a part, leading to a collectively processed final inference result and improved resilience and dynamic resource optimization.

The following figure shows the case where the local machine that requests AI service to horizontal multiple cloud servers.

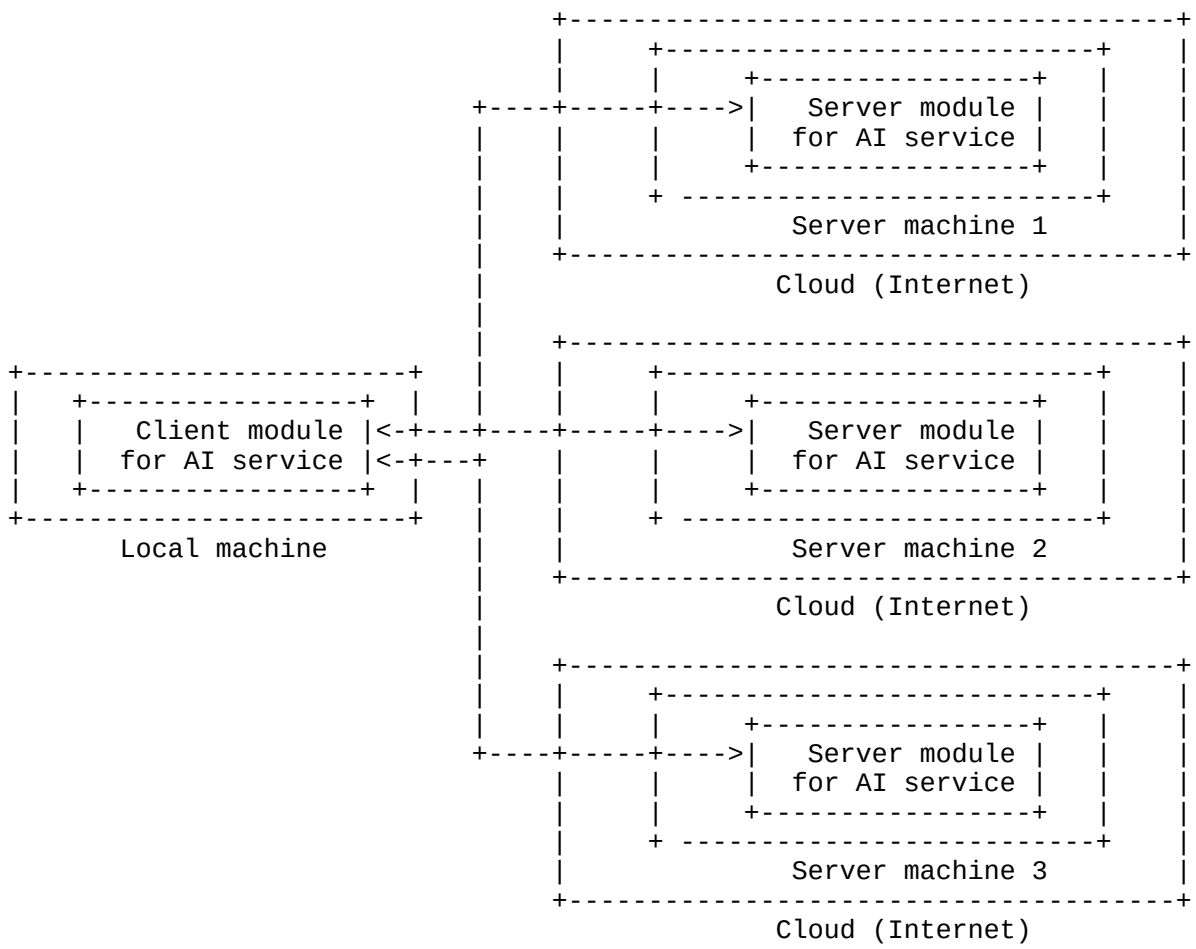

Figure 6: AI inference service on horizontal multiple servers

3.6. Network-side utilization for AI learning

Collecting and preprocessing of data and training an AI model requires a high-performance resource such as CPU, GPU, Power, and Storage. To mitigate this requirement, we can utilize a network-side configuration. Typically, federating learning is a machine learning technique that trains an AI model across multiple decentralized servers. It is a contrast to traditional centralized machine learning techniques where all the local datasets are uploaded to one server. In this federated learning, it enables multiple network nodes to build a common machine learning model. These network nodes

or servers can be located in the same data center network or multiple data center network.

And, transfer learning is a machine learning technique that focuses on storing information gained while solving one problem and applying it to a different but related problem. In this transfer learning, we can utilize a network configuration to transfer common information and knowledge between different network nodes within one data center network or across multiple data center networks.

4. Considerations of network/system for AI services

As described in the previous chapter, the AI server module that directly performs AI inference services by utilizing AI models can be performed on a local machine or a cloud server or an edge device.

In theory, if AI inference service is performed on a local machine, AI service can be provided without communication delay time or packet loss, but a certain amount of hardware performance is required to perform AI service inference. So, in the future environment where AI services become popular, such as when various AI services are activated and AI services are disseminated, the cost of a machine that performs AI services is important

If so, whether the AI inference service will be performed on the cloud server or the discount price on the edge device can be a determining factor in the system configuration.

4.1. Considerations of the functional characteristics of the hardware

When AI inference service request is made to a distant cloud server, it may take a lot of time to transmit, but it has the advantage of being able to perform many AI inference service requests in a short time, and the accuracy of AI service inference increases. Conversely, when an AI service request is made to a nearby edge device, the transmission time is short, but many AI inference service requests cannot be performed at once, and the accuracy of AI service inference is lowered.

Therefore, by analyzing the characteristics and requirements of the AI service to be performed, it is necessary to determine where to perform the AI inference service on a local machine, a cloud server, or an edge device.

The hardware characteristics of the machine performing the AI service varies. In general, machines on cloud servers are viewed as machines with higher performance than edge devices. However, the performance of AI inference service varies depending on how the hardware such as

CPU, RAM, GPU, and network interface is configured for each cloud server and edge device. If we do not think about cost, it is good to configure a system for performing AI services with a machine with the best hardware performance, but in reality, we should always consider the cost when configuring the system. So, according to the characteristics and requirements of the AI service to be performed, the performance of the local machine, cloud server, and edge device must be determined.

Performance evaluation is possible through the performance matrix presented in the standard of ETSI[MEC.IEG006]. The performance metrics suggested by the ETSI standard are as follows. These metrics is divided into two groups, namely Functional metrics, which assess the user performance and include some classical indexes such as latency in task execution, device energy efficiency, bit-rate, loss rate, jitter, Quality of Service (QoS), etc.; and Non-functional metrics that, instead, focus on the MEC(Mobile Edge Computing) network deployment and management. Non-functional metrics include the following indexes. Service life-cycle(instantiation, service deployment, service provisioning, service update (e.g. service scalability and elasticity), service disposal), service availability and fault tolerance (aka reliability), service processing/ computational load, global mobile equipment host load, number of API request (more generally number of events) processed/second on mobile equipment host, delay to process API request (north and south), number of failed API request. The sum of service instantiation, service deployment, and service provisioning provide service boot-time.

4.2. Considerations for the characteristics of the AI model

According to the characteristics of the AI service, although not directly related to communication/network, the biggest influence on AI inference services is the AI model to be used for AI inference service. For example, in AI services such as image classification, there are various types of AI models such as ResNet, EfficientNet, VGG, and Inception. These AI models differ in AI inference accuracy, but also in AI model file size and AI inference time. AI models with the highest inference accuracy typically have very large file sizes and take a lot of AI inference time. So, when constructing an AI service system, it is not always good to choose an AI model with the highest AI inference accuracy. Again, it is important to select an AI model according to the characteristics and requirements of the AI service to be performed.

Experimentally, it is recommended to use an AI model with high AI inference accuracy in the cloud server, and use an AI model that can provide fast AI inference service although the AI inference accuracy is slightly lower for the fast AI inference service in the edge device.

It might be a bit of an implementation issue, but we should also consider how we deliver AI services on cloud servers or edge devices. With the current technology, a traditional web server method or a server method specialized for AI service inference (e.g., Google's Tensorflow Serving) can be used. Traditional web server methods such as Flask and Django have the advantage of running on various types of machines, but since they are designed to support general web services, the service execution time is not fast. Tensorflow Serving uses the features of Tensorflow to make AI service inference services very fast and efficient. However, older CPUs that do not support AVX cannot use the Tensorflow serving function because Google's Tensorflow does not run. Therefore, rather than unconditionally using the server method specialized in AI service inference, it is necessary to decide the AI server module method that provides AI services in consideration of the hardware characteristics of the AI system that can be built.

4.3. Considerations for the characteristics of the communication method

The communication method for transferring data to request AI inference service is also an important decision in constructing an AI system. Using the traditional REST method, it can be used for various machines and services, but its performance is inferior to gRPC. There are many advantages to using gRPC for AI inference services, as gRPC uses HTTP/2 and protocol buffers for communication, providing low latency and efficient data serialization, and enabling large capacity and efficient data transfer compared to REST. Alternatively, the QUIC protocol or other future transport protocol can also be used to request an AI inference service. Which transport protocol is used is beyond the scope of this document.

Cloud-edge-endpoint collaboration-based AI service development is actively underway. In particular, in the case of AI services that are sensitive to network delays, such as object recognition and autonomous vehicle services, (micro)services for inference are placed on edge devices close to the client to obtain fast inference results and provide services. As such, in the development of intelligent IoT services, various devices that can provide computing services within the network, such as edge devices, are being added as network elements, and the number of IoT devices using them is rapidly increasing. Therefore, a new function for computing resource management and operation is required in terms of providing computing

services within the network. In addition, to operate distributed AI service on network, the network policy for collaboration is required between edge devices or between edge device and endpoint device that support computing resource for AI service.

In network policy, in order to efficiently support distributed AI services, existing networks must be provided with the collaboration of AI service between edge devices such as multi-edge network configuration for AI service aware traffic steering in multi-edge network, so that the client can receive distributed AI service efficiently in various network environments and the multi-edge network configuration enables dynamically vary network resource and the computing resource of edge device. For example, AI tasks message exchanges must be possible between edge devices or between edge devices and endpoint devices to provide collective AI inference. Another example is to place caching at the edge devices between the client and the central cloud to allow make a copy of an accessed/ requested file at the edge devices, which are then served for subsequent requests from the same client or other clients. Also, there are various delay sensitive AI services based on edge device in the network. They are divided into in-time delay sensitive AI services with a deadline limit and on-time delay sensitive AI services with the set of a time-range. In particular on-time delay sensitive AI services want to return the results of prediction within the time range. Therefore, distributed AI service must be able to provide proper AI service in terms of the delay performance for distributed AI service through network. Therefore, the client of AI service should be able to be provided both in-time and on-time delay sensitive services and be interacted with edge devices where distributed AI service is built.

5. Addressing challenges for coupling AI and NM

The document [I-D.irtf-nmrg-ai-challenges] describes the challenges found in the application of AI to network management problems. They are separated into low-level and high-level challenges. This section describes how the concepts discussed in the present document are linked to those challenges.

Some of the links between the structures presented in this document and AI challenges involve the alignment with AINEMA [I-D.pedro-nmrg-ai-framework].

AINEMA [I-D.pedro-nmrg-ai-framework] is a framework of functions required for exploiting AI services and their interconnection. Intelligent management systems based on AINEMA, as shown in [TNSM-2018], introduce AI functions in the network management cycles, and are able to express management decisions in terms of network intents, as described in [I-D.pedro-ite].

5.1. Low-level challenges

The first challenge (C1) concerns to the combinatorial explosion of the solution space in relation to the size of the problem. The present document proposes to tackle part of such complexity by separating the AI work in multiple elements of the network. The separation is asymmetrical, so that some elements (cloud side) will contribute more computation power to the distributed service.

The second challenge (C2) regards the dimensional and context uncertainty and unpredictability. The present document focus on cloistering the AI models and data within pre-defined client-edge-cloud-server structures.

The third challenge (C3) is to ensure that AI to is able to provide prompt responses and decisions to management questions. The key aspect discussed in this document to resolve this challenge is the integration of end-node, in-network, edge, an cloud computing services. This allows AI computations to be performed in the best place possible. Time elapsed to get fast responses of easy operations will be minimized by executing them in the end-nodes (e.g., vehicles) and edge devices, while the time needed to compute more complex operations is minimized by offloading them to cloud computing services.

The fourth challenge (C4) states the difficulties related to resolving data imperfections and scalability of techniques. This challenge is specific to each information domain and problem. However, as presented in this document, the mentioned difficulties in C4 is relieved by exposing AINEMA [I-D.pedro-nmrg-ai-framework] functions instantiated in cloud continuum to the AI services, so that AI services transparently gain capabilities such as homogenization and scalability. Instantiating a distributed AI system in a cloud continuum enables the AI system to have many functions to deal with data homogenization, resolving high data rates, etc.

The fifth challenge (C5) concerns the integration of AI services with existing automation and human processes. The flexibility of the structures presented in this document allow them to be connected to existing systems, being aligned and somewhat interconnected with AINEMA [I-D.pedro-nmrg-ai-framework]. The result is exemplified in the support for network digital twin and vehicle environments.

The sixth challenge (C6) exposes the need for cost-effective solutions. Enabling AI services to be deployed in the cloud continuum supports the maximization of effectiveness by dynamically relocating services to the cheapest provider point that is able to accomplish the computation requirements. This relocation is guided by intents, as described in [I-D.pedro-ite], and implemented by an independent management cycle, as described in AINEMA [I-D.pedro-nmrg-ai-framework] and [TNSM-2018]. The result is that the cost of a distributed AI system is continuously being checked and adjusted by requesting the underlying infrastructure to modify the point of instantiation of each part of the AI system.

5.2. High-level challenges

The first challenge (H1) relies in the observation that AI techniques were developed in a different area---imaging---, so they must be extensively tailored for network problems. Although this is a quite complex challenge, this document supports its resolution by enabling AI models and algorithms to evolve separately. Multiple providers will offer multiple solutions and they will be stitched together by following the structures introduced above, as well as the functions provided by the interconnection and alignment with AINEMA [I-D.pedro-nmrg-ai-framework].

The second challenge (H2) conveys the mismatch that exists from the original data and internal data used in AI models. Although the present document does cover this mismatch, the structures presented in this document help alleviate the burden by relocating some AI processes as close as possible to the end-points (e.g., vehicles and NDT). The same structures support the implementation of measures to protect privacy. The AI services that are closer to the edges will deal with sensitive data and ensure privacy protection by, for example, anonymizing it. Those services are instantiated within the data domain, so private data does not crosses the boundaries of its data domain.

The third challenge (H3) consists on the level of acceptance that an AI system experiences from administrators and operators. It is agreed that giving full control of AI operations to administrators and operators increases such level of acceptance. The structures presented in this document support the involvement of administrators

and operators in AI system processes through the provision of policies and network intents. On the one hand, aligning the distributed AI system specified in this document with AINEMA [I-D.pedro-nmrg-ai-framework], enables the enforcement high-level policies and management goals provided by administrators and operators. On the other hand, aligning the distributed AI system specified in this document with intent-based networking systems, particularly intent translation [I-D.pedro-ite], enables administrators and operators to use high-level specifications (namely network intents) to communicate the policies, requirements, and goals that must be enforced to the AI system constructed over the cloud continuum.

6. Use cases of deploying network-based AI services

6.1. Deploying AI services for self-driving vehicles

Various sensors are used in self-driving vehicles, and the final judgment is made by combining these data. Among them, camera data-based object detection solves parts that expensive equipment such as LiDAR and RADAR cannot solve. Camera-based object detection performs various tasks, and in addition to lane recognition for maintaining driving lanes and changing lanes, it also supports safe driving and parking assistance by distinguishing shape information such as pedestrians, signs, and parking vehicles along the road.

In order to perform such driving assistance and autonomous driving, object detection needs to be performed in real time. The minimum FPS(Frames Per Second) to be considered real-time in autonomous driving is 30 FPS[Object_detection]. No matter how high the accuracy is, it cannot be used for autonomous driving if it does not meet the corresponding reference value.

Task offloading refers to a technology or structure that transfers computing tasks to other processing devices or systems to perform them. Task offloading can quickly process tasks that exceed the performance limits of devices that lack resources by delivering tasks from devices with limited computing power, storage space, and power to devices that are rich in computing resources.

For devices with low hardware performance (e.g., NVIDIA Jetson Nano board, Qual-core ARM A57, 4GB RAM), all locally without task offloading results in 4.6 FPS, which is difficult to perform object detection-based autonomous driving. On the other hand, if task offloading is applied to perform object detection on devices with high hardware performance (e.g., Intel i7, RTX 3060, 32GB RAM) and the rest of the work is performed on the client, 41.8 FPS will be obtained. This is a result that satisfies 30 FPS, which is the reference FPS of object detection-based autonomous driving.

In the case of AI services such as object detection, if it is difficult to perform on resource-constrained devices, it can be seen that the task offloading structure shows some efficiency. However, without performing all operations locally, task offloading operations between network nodes can affect the entire time because the larger the size of the data, the greater the communication latency. Therefore, in such a network distributed environment, the provision of AI services should be designed in consideration of various variables. The Figure 7 shows an example of distributed AI deployment in a self-driving vehicle when it does not have enough capabilities to proceed the object detection operation in real-time and it asks some tasks to edge devices and cloud servers.

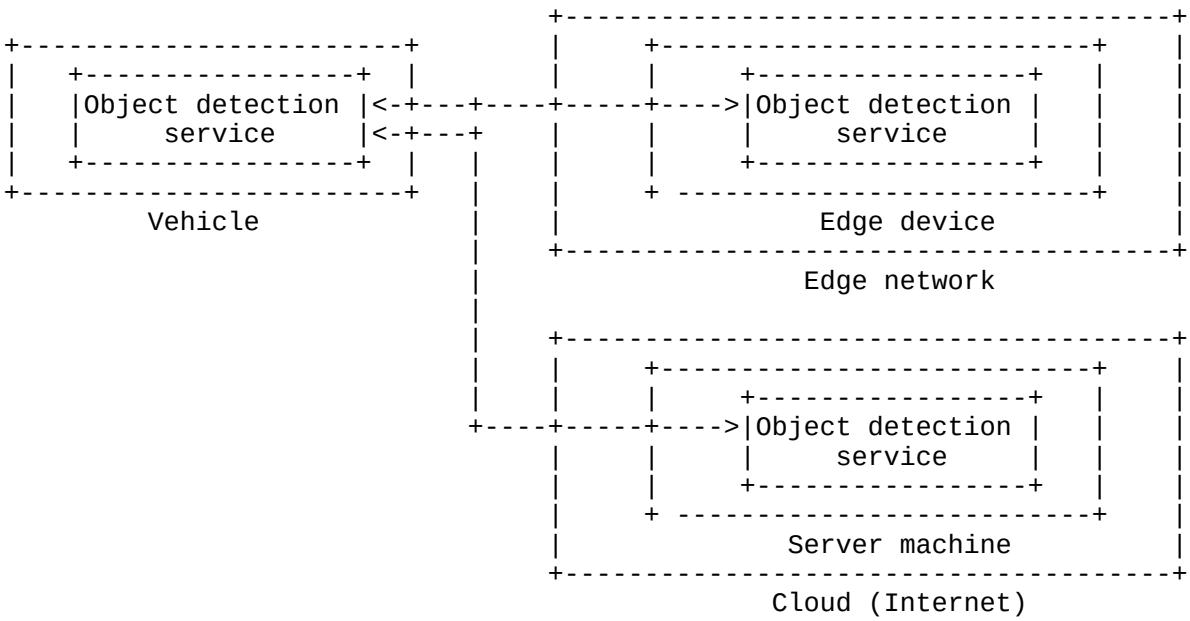

Figure 7: Distributed object detection service in self-driving vehicle

6.2. Deploying AI services for network digital twins

Network digital twin also need to build distributed AI services. The purpose of a network digital twin is described in [I-D.irtf-nmrg-network-digital-twin-arch]. In particular, the network digital twin provides network operators with technology that enables data driven network management and allows real time interaction between physical network and twin network. To achieve this, the network digital twin will use AI capabilities for various purposes such as scenario planning, impact analysis and change management.

Various AI functions will be applied for optimal network operation and management. However, the actual physical network consists of various different network devices and has a complex structure for various different type of data such as topology data, configuration data, state data. In addition, in a large-scale network environment, the network overhead is very large and expensive to collect and store information from many network devices in a centralized manner, and to create and change network management policies based on it.

Therefore, there is a need for a method to apply AI functions based on a distributed form for network operation and management. In particular, the actual physical network structure is built in a logical hierarchical structure. Therefore, it is necessary to apply a distributed AI method that considers the logical hierarchical network structure environment.

In order to optimally perform network operation and management through distributed AI methods, it is necessary to generate AI function-based network operation and management topology model and policy models and an operational method to distribute the generated AI function-based network policies across different networks or administrative domains. In particular, in order to operate a network digital twin in a large-scale network environment, it is necessary to generate AI-based network policy models in a distributed manner. A federated learning algorithm or a transfer learning algorithm that can learn large-scale networks in a distributed manner can be applied.

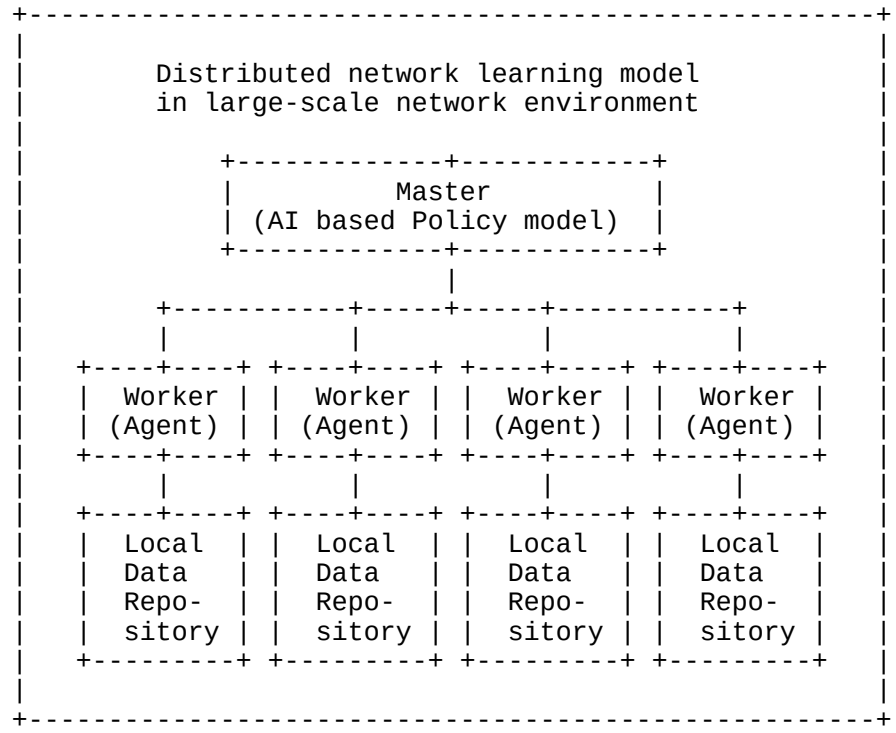

Figure 8: Distributed learning model of network learning for network digital twins

As shown in Figure 8, in order to learn a large-scale network through a distributed learning method, a local data repository to store network data must be established in each region, for example, based on location or AS (Autonomous System). Therefore, the distributed learning method learns through each worker (agent) based on the local network data stored in the local network data repository, and generates a large-scale network policy model through the master. This distributed learning method can reduce the network overhead of centralized data collection and storage, and reduce the time required to create AI models for network operation and management policies for large-scale networks. In addition, the network policy model generated by the worker can be used as a locally optimized network policy model to provide AI-based network operation and management policy services optimized for local network operations.

The distributed deployment of trained AI network policy models can be deployed on network devices that can manage and operate the local network to minimize network data movement. For example, in a large-scale network consisting of multiple ASes, AI network policy models can be deployed per AS to optimize network operation and management. Figure 9 shows an example of operating and managing a network by distributing AI network policy models by AS.

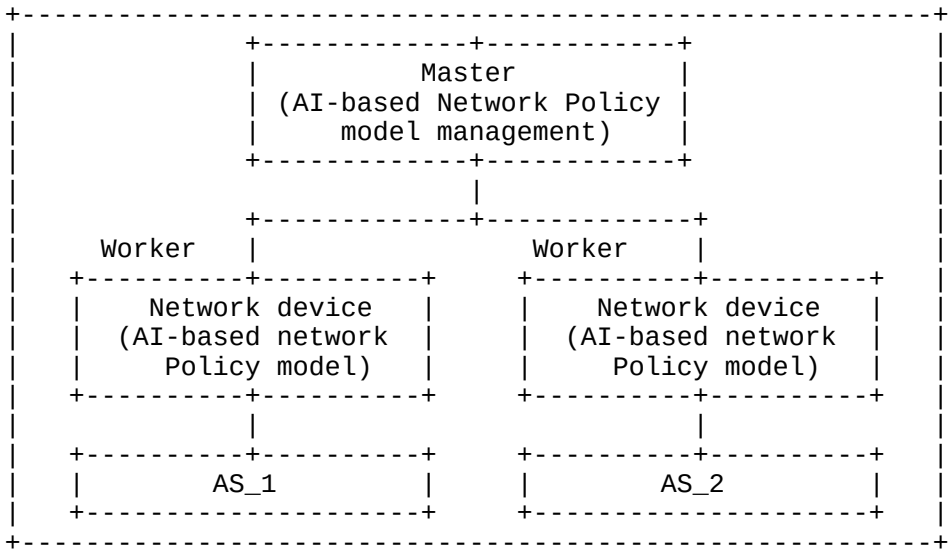

Figure 9: Distributed deployment of trained AI network policy models

7. IANA Considerations

There are no IANA considerations related to this document.

8. Security Considerations

When AI service is performed on a local machine, there is no security issue, but when AI service is provided through a cloud server or edge device, IP address and port number may be known to the outside can attack. Therefore, when providing AI services by utilizing machines on the network such as cloud servers and edge devices, it is necessary to analyze the characteristics of the modules to be used well, identify vulnerabilities in security, and take countermeasures.

9. Acknowledgements

TBA

10. References

10.1. Normative References

[RFC6574] Tschofenig, H. and J. Arkko, "Report from the Smart Object Workshop", RFC 6574, DOI 10.17487/RFC6574, April 2012, <https://www.rfc-editor.org/info/rfc6574>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <https://www.rfc-editor.org/info/rfc7452>.

[RFC9556] Hong, J., Hong, Y., de Foy, X., Kovatsch, M., Schooler, E., and D. Kutscher, "Internet of Things (IoT) Edge Challenges and Functions", RFC 9556, DOI 10.17487/RFC9556, April 2024, <https://www.rfc-editor.org/info/rfc9556>.

10.2. Informative References

[I-D.irtf-nmrg-network-digital-twin-arch] Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture", Work in Progress, Internet-Draft, draft-irtf-nmrg-network-digital-twin-arch-11, 6 July 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-network-digital-twin-arch-11>.

[I-D.irtf-nmrg-ai-challenges] Fran巽ois, J., Clemm, A., Papadimitriou, D., Fernandes, S., and S. Schneider, "Research Challenges in Coupling Artificial Intelligence and Network Management", Work in Progress, Internet-Draft, draft-irtf-nmrg-ai-challenges-05, 18 March 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-ai-challenges-05>.

[I-D.pedro-nmrg-ai-framework] Martinez-Julia, P., Homma, S., and D. Lopez, "Artificial Intelligence Framework for Network Management", Work in Progress, Internet-Draft, draft-pedro-nmrg-ai-framework-05, 20 October 2024, <https://datatracker.ietf.org/doc/html/draft-pedro-nmrg-ai-framework-05>.

[I-D.pedro-ite]
Martinez-Julia, P. and J. P. Jeong, "Intent Translation Engine for Intent-Based Networking", Work in Progress, Internet-Draft, draft-pedro-ite-01, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-pedro-ite-01>.

[CG-AIoT] "ITU-T CG-AIoT", <https://www.itu.int/en/ITU-T/studygroups/2017-2020/20/Pages/ifa-structure.aspx>.

[tinyML] "tinyML Foundation", <https://www.tinyml.org/>.

[AI_inference_archtecture]
"IBM Systems, AI Infrastructure Reference Architecture", <https://www.ibm.com/downloads/cas/W1JQBNJV>.

[Google_cloud_iot]
"Bringing intelligence to the edge with Cloud IoT", <https://cloud.google.com/blog/products/gcp/bringing-intelligence-edge-cloud-iot>.

[MEC.IEG006]
ETSI, "Mobile Edge Computing; Market Acceleration; MEC Metrics Best Practice and Guidelines", Group Specification ETSI GS MEC-IEG 006 V1.1.1 (2017-01), January 2017.

[Object_detection]
Lewis, "Object Detection for Autonomous Vehicles Gene", 2016.

[TNSM-2018]
P. Martinez-Julia, V. P. Kafle, and H. Harai, "Exploiting External Events for Resource Adaptation in Virtual Computer and Network Systems, in IEEE Transactions on Network and Service Management. Vol. 15, n. 2, pp. 555--566, 2018.", 2018.

Authors' Addresses

Yong-Geun Hong
Daejeon University
62 Daehak-ro, Dong-gu
Daejeon
34520
South Korea
Phone: +82 42 280 4841
Email: yonggeun.hong@gmail.com

Joo-Sang Youn
DONG-EUI University
176 Eomgwangno Busan_jin_gu
Busan
614-714
South Korea
Phone: +82 51 890 1993
Email: joosang.youn@gmail.com

Seung-Woo Hong
ETRI
218 Gajeong-ro Yuseong-gu
Daejeon
34129
South Korea
Phone: +82 42 860 1041
Email: swhong@etri.re.kr

Pedro Martinez-Julia
National Institute of Information and Communications Technology
4-2-1, Nukui-Kitamachi, Koganei, Tokyo
184-8795
Japan
Phone: +81 42 327 7293
Email: pedro@nict.go.jp

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
210012
China
Email: bill.wu@huawei.com