

**Draft new Recommendation ITU-T Y.3186 (formerly Y.IMT2020-DJLML)**

**Requirements and framework for distributed joint learning to enable machine learning in future networks including IMT-2020**

**Summary**

The Recommendation specifies scenarios, requirements, framework and flow diagram for distributed joint learning to enable machine learning in future networks including IMT-2020. The Recommendation can help to realize a highly automated, intelligent, and multi-party collaborative network.

**Keywords**

Distributed joint learning, future network, machine learning

## Table of Contents

	<b>Page</b>
1. Scope.....	3
2. References.....	3
3. Definitions.....	4
3.1. Terms defined elsewhere .....	4
3.2. Terms defined in this Recommendation .....	4
4. Abbreviations and acronyms.....	4
5. Conventions .....	5
6. Overview .....	5
7. Scenarios of DJL in future networks including IMT-2020.....	6
7.1. Scenarios from the perspective of the data source of participating DJL nodes.....	6
7.2. Scenarios from the perspective of data distribution in DJL .....	8
8. Requirements of DJL in future networks including IMT-2020 .....	10
8.1 DJL task management.....	10
8.2 DJL node management .....	10
8.3 DJL resource management .....	11
8.4 DJL model management.....	11
8.5 DJL data management .....	12
8.6 DJL interfaces.....	12
8.7 DJL connectivity.....	12
8.8 DJL reliability management .....	12
8.9 DJL security and privacy protection.....	13
9. Framework of DJL in future networks including IMT-2020.....	13
9.1. The architectural components for DJL .....	13
9.2. Architectural framework.....	16
9.3. Deployment options of the DJL components .....	17
9.4. DJL data transmission optimization .....	17
10. Flow diagram for DJL.....	18
11. Security considerations .....	19
Bibliography.....	19

# Draft new Recommendation ITU-T Y.3186 (formerly Y.IMT2020-DJLML)

## Requirements and framework for distributed joint learning to enable machine learning in future networks including IMT-2020

### 1. Scope

This Recommendation specifies requirements and framework for distributed joint learning to enable machine learning in future networks including IMT-2020. The Recommendation addresses the following topics concerning distributed joint learning to enable machine learning in future networks including IMT-2020:

- Scenarios;
- Requirements;
- Framework;
- Flow diagram.

### 2. References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T F.747.13 ]	Recommendation ITU-T F.747.13 (2023), <i>Requirements and reference framework of cloud-edge collaboration in industrial machine vision systems</i>
[ITU-T Y.3104]	Recommendation ITU-T Y.3104 (2018), <i>Architecture of the IMT-2020</i>
[ITU-T Y.3115]	Recommendation ITU-T Y.3115 (2022) <i>AI integrated cross-domain network architecture for future networks including IMT-2020 network</i>
[ITU-T Y.3172]	Recommendation ITU-T Y.3172 (2019), <i>Architectural framework for machine learning in future networks including IMT-2020.</i>
[ITU-T Y.3174]	Recommendation ITU-T Y.3174 (2020), <i>Framework for data handling to enable machine learning in future networks including IMT-2020.</i>
[ITU-T Y.3179]	Recommendation ITU-T Y.3179 (2021), <i>Architectural framework for machine learning model serving in future networks including IMT-2020</i>
[ITU-T Q.1742.3]	Recommendation ITU-T Q.1742.3 (2004), <i>IMT-2000 references (approved as of 30 June 2003) to ANSI-41 evolved core network with cdma2000 access network</i>

### 3. Definitions

#### 3.1. Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1 machine learning (ML) [ITU-T Y.3172]:** Processes that enable computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed.

NOTE 1 – This definition is adapted from [b-ETSI GR ENI 004].

NOTE 2 – Supervised machine learning and unsupervised machine learning are two examples of machine learning types.

**3.1.2 machine learning function orchestrator (MLFO) [ITU-T Y.3172]:** A logical node with functionalities that manage and orchestrate the nodes in a machine learning pipeline.

**3.1.3 machine learning overlay (ML overlay) [ITU-T Y.3172]:** A loosely coupled deployment model of machine learning functionalities whose integration and management with network functions are standardised.

**3.1.4 machine learning pipeline (ML pipeline) [ITU-T Y.3172]:** A set of logical nodes, each with specific functionalities, that can be combined to form a machine learning application in a telecommunication network.

**3.1.5 machine learning underlay network (ML underlay network) [ITU-T Y.3172]:** A telecommunication network and its related network functions which interfaces with corresponding machine learning overlays.

NOTE – An IMT-2020 network is an example of a machine learning underlay network.

**3.1.6 IP network [ITU-T Q.1742.3]:** The IP network corresponds to IP-based packet data networks that provide a transport mechanism between the core network and external IP networks. IP Network represents packet networks connected to the core network including the public Internet, private IP backbone networks and private IP networks such as a corporate Intranets.

**3.1.7 federated machine learning (FML) [ITU-T F.747.13]:** Federated machine learning is a framework or system that enables multiple participants to collaboratively build and use machine learning models without disclosing the raw and private data owned by the participants while achieving good performance.

#### 3.2. Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1 distributed joint learning (DJL):** Also called decentralized learning, it refers to techniques that use multiple computing nodes for machine learning. NOTE - This approach deals with data and parameters located at the edge. It includes, but is not limited to, federated learning, partitioned learning and distributed reinforcement learning.

### 4. Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AI Artificial Intelligence

CPU	Central Processing Unit
DJL	Distributed Joint Learning
FML	Federated Machine Learning
FPGA	Field Programmable Gate Arrays
FTL	Federated Transfer Learning
GPU	Graphics Processing Unit
HFL	Horizontal Federated Learning
IT	Information Technology
ML	Machine Learning
PCF	Policy Control Function
QoS	Quality of Service
UE	User Equipment
V2X	Vehicle to Everything
VFL	Vertical Federated Learning

## 5. Conventions

In this Recommendation:

The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.

The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option, and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.

In this Recommendation, "distributed joint learning to enable machine learning" is also named as "distributed joint learning (DJL)" for simplicity.

## 6. Overview

The in-depth development of intelligence in the network field is playing a vital role in improving network efficiency and optimizing network operations and maintenance. When collecting and training data from various sources, new challenges have to be considered, in particular the issues of data islands, data privacy as well as the efficiency of data and model computing. This has a significant impact on artificial intelligence (AI)/machine learning (ML) technologies that require big data analysis.

Concerning intelligent closed loops in the IMT-2020 network, closed loop functionalities are used inside a network domain or cross-domains by the means of collaboration among network domains, as specified in [ITU-T Y.3115]. However, it is difficult for a single domain or cross-domains to centralize the raw data that are distributed in different areas. It is desirable for a single domain or cross-domains to share ML model or data analytics in a closed loop. This makes necessary to introduce a DJL based approach to address this issue.

DJL capabilities, dealing with both data and parameters located at the edge, go beyond the capabilities enabled by the federated learning based approach, by supporting other capabilities to address computing requirements.

In this Recommendation the approach of DJL is introduced to enable machine learning into future networks including IMT-2020. The study can help to realize a highly automated, intelligent, and multi-party collaborative network.

**7. Scenarios of DJL in future networks including IMT-2020**

Comprehensive data is very important for training a high-quality AI model. But the data in most fields or areas may be limited or of poor quality. However, due to privacy protection or a large amount of data transmission, it is difficult to fuse the data together in a common site by transporting the data across organizations. So DJL is a suitable solution to break the barriers between data sources. The DJL participants with different data sources do not need to directly exchange their raw data, and only need to exchange the intermediate results of model training so as to achieve distributed joint training of models based on different data sources.

NOTE – The following clauses 7.1 and 7.2 illustrate scenarios – from two different perspectives – referring to the IMT-2020 network, but are expected to be applicable also to future networks.

**7.1. Scenarios from the perspective of the data source of participating DJL nodes**

Multiple data sources participate in DJL. As shown in figure 7-1, the data used in DJL may come from the IMT-2020 network and other sources.

The data from the IMT-2020 network is generated in network nodes. A network node may be part of a cluster, a cluster may be part of a region, a region may be part of an IMT-2020 network, and different IMT-2020 networks may belong to different network operators. The different network nodes may belong to the same cluster or different clusters, or may belong to the same region or different regions, or may belong to the same IMT-2020 network or different IMT-2020 networks. Other data may come from UEs and AFs (e.g., V2X application servers, IT systems of external enterprises, network functions from other network operators).

NOTE 1 – A cluster is a collection of the IMT-2020 network nodes that are grouped for a particular purpose. For example, the network nodes of an access network may be grouped into a cluster.

NOTE 2 – A region indicates a geographical location, and an IMT-2020 network may span one or more regions.

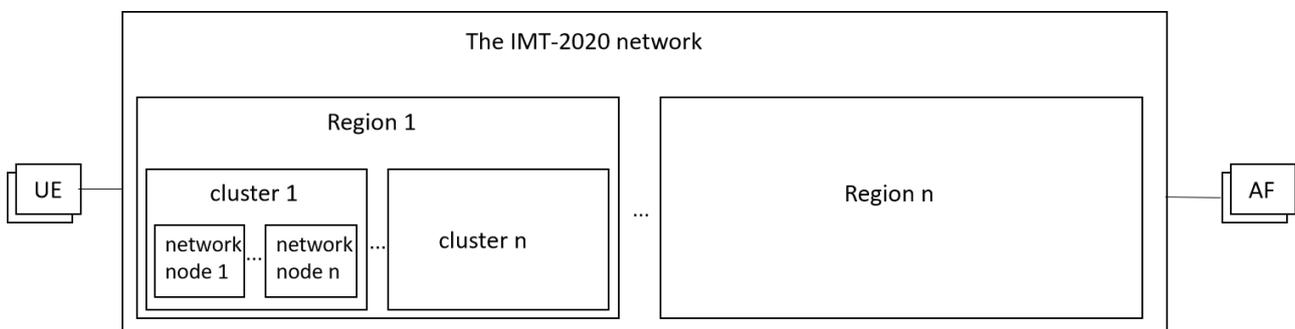


Figure7-1: The distribution of data sources in joint learning

**7.1.1. The IMT-2020 network enables DJL with the use of network data**

DJL can be used to aggregate network data in the IMT-2020 network in order to train the AI/ML models which can be used in the IMT-2020 network.

The scenarios of DJL with respect to the use of network data are as follows: DJL on data from

different regions in the IMT-2020 network (sub-scenario 1), DJL on data from different clusters of the same region (sub-scenario 2), DJL on data from different network nodes in the same cluster (sub-scenario 3).

As far as sub-scenario 1 is concerned, data of different network nodes in different regions can be aggregated and applied through DJL for AI/ML applications. As shown in figure 7-2, the grey nodes are data source examples for the scenario.

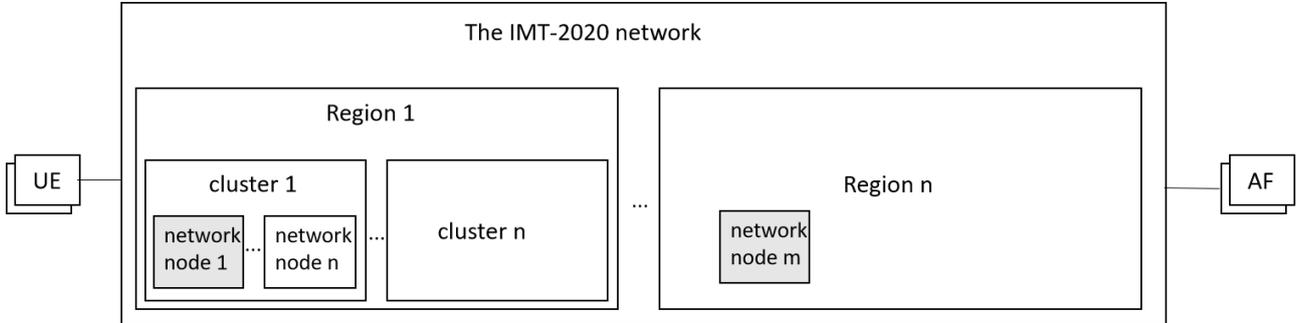


Figure7-2: Different network nodes in different regions of the same operator’s network as data sources for DJL

As far as sub-scenario 2 is concerned, data of different network nodes in the different clusters of the same region can be aggregated and applied through DJL for AI/ML applications. As shown in figure 7-3, the grey nodes are data source examples for the scenario.

NOTE 1 – For example, data from access network nodes, core network nodes, and transport network nodes can be aggregated and applied through DJL.

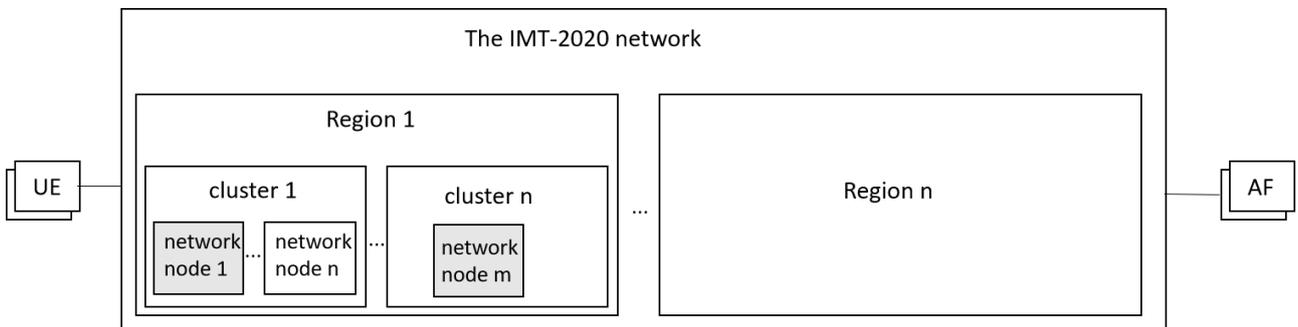


Figure7-3: Different network nodes in the different clusters of the same region as data sources for DJL

As far as sub-scenario 3 is concerned, data of different network nodes in the same cluster can be aggregated and applied through DJL for AI/ML applications. As shown in figure 7-4, the grey nodes are data source examples for the scenario.

NOTE 2 – For example, DJL can be applied among distributed UPF network nodes in the same cluster.

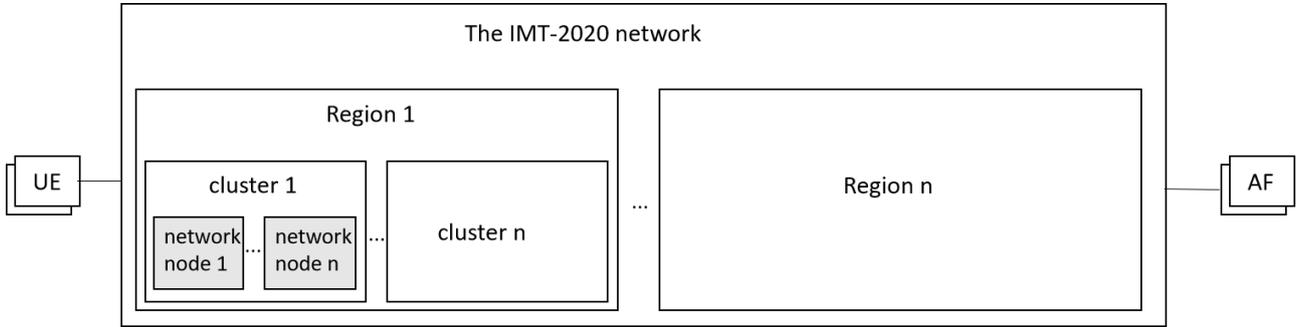


Figure7-4: Different network nodes in the same cluster as data sources for DJL

**7.1.2 The IMT-2020 network enables DJL with the use of network data and external data**

DJL can be used to aggregate network data and external network data in the IMT-2020 network in order to train the AI/ML models which can be used in the IMT-2020 network.

The data from other sources can be UEs or AFs. As shown in figure 7-5, the grey nodes are data source examples for the scenario.

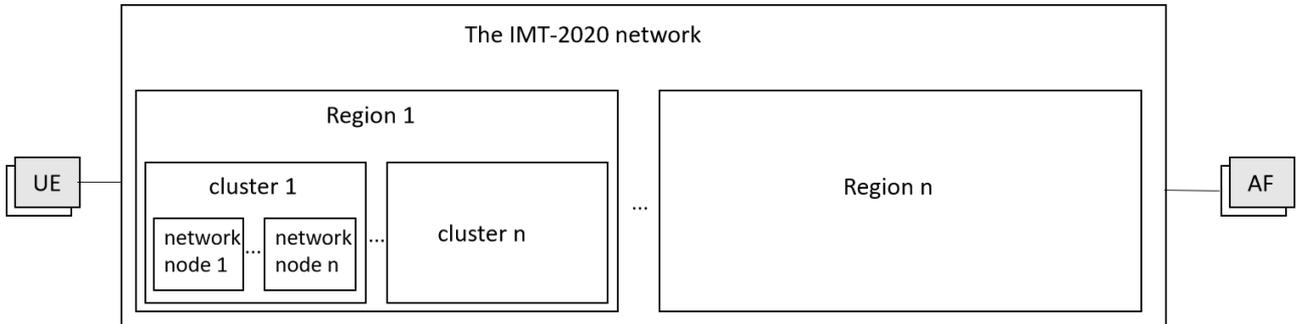


Figure7-5: IMT-2020 network nodes as data sources and UEs or AFs as other data sources for DJL

NOTE – V2X scenario [b-3GPP TS 22.186] is a typical use case, in which AF corresponds to the V2X application server function and UE corresponds to a V2X terminal.

**7.2. Scenarios from the perspective of data distribution in DJL**

The operations of vertical federated learning, horizontal federated learning and federated transfer learning for data sources are shown in, respectively, figures 7-2a, 7.2b and 7.2c.

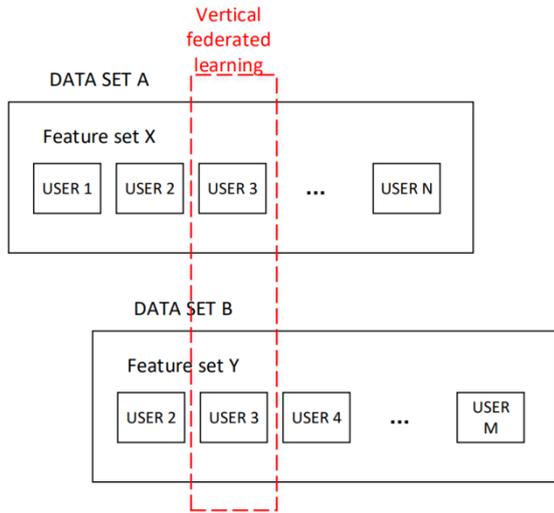


Figure 7-2a Vertical federated learning

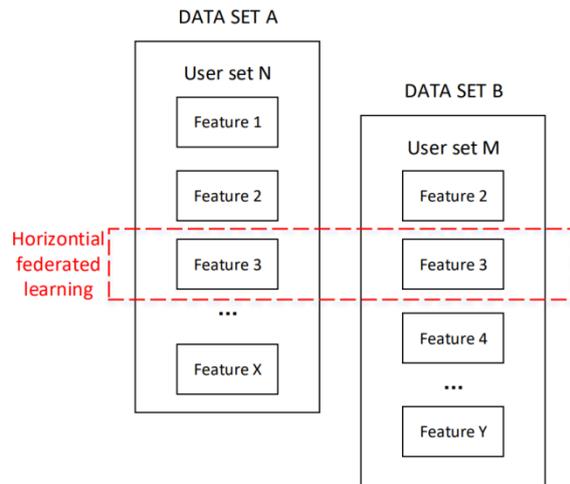


Figure 7-2b Horizontal federated learning

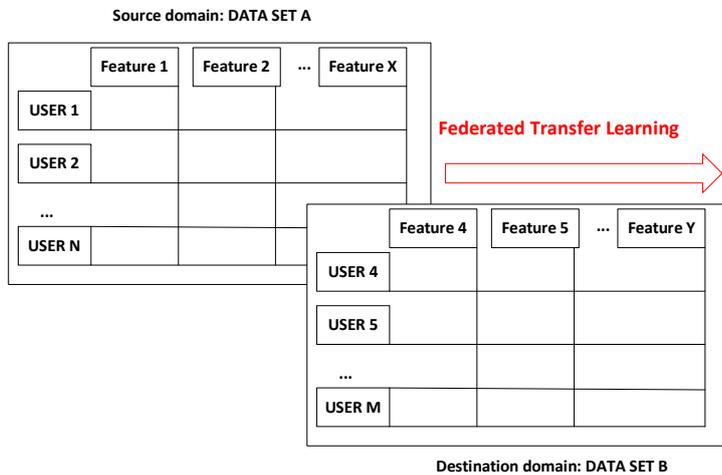


Figure 7-2c Federated transfer learning

As shown in Figure 7-2a, vertical federated learning (VFL) - also named as feature-based federated learning - is a type of federated machine learning (FML) [ITU-T F.747.13] suitable for the scenarios where data is vertically distributed. NOTE 1 - The two data sets shown in figure 7-2a share the same user ID space but differ in feature space.

NOTE 2 – “Feature” (as used in ML) represents a subset of measurable properties of data items.

As shown in Figure 7-2b, horizontal federated learning (HFL) is a type of FML suitable for the scenarios where data is horizontally distributed. NOTE 3 - The two data sets shown in figure 7-2b share the same feature space but differ in user ID space.

As shown in Figure 7-2c, federated transfer learning (FTL) is a type of FML suitable for the scenarios where data is heterogeneously distributed.

NOTE 4 - The two data sets shown in figure 7-2c differ in both user ID space and feature space.

NOTE 5 - FTL takes advantage of transfer learning techniques to provide solutions for different user ID and feature spaces under a federation. If we consider DATA SET A representing the source domain and DATA SET B representing the destination domain, federated transfer learning learns the distribution of features in the source domain and migrates the feature information of the source domain to the destination domain, while the local data does not leave the local area during this

migration process.

The IMT-2020 network may enable DJL with the use of vertical federated learning.

Data generated by a user (same user ID) in the IMT-2020 network with different features (i.e., from different sources) can be collected, analyzed, and used to train a model that can get more comprehensive preferences and characteristics of the user.

The IMT-2020 network may enable DJL with the use of horizontal federated learning.

Data generated by different users (different user IDs) in the IMT-2020 network with the same feature can be collected and analyzed, and used to train a model that can be more appropriate to characterize these users.

The IMT-2020 network may enable DJL with the use of federated transfer learning. Data generated by different users (different user IDs) in the IMT-2020 network with different features can be collected, analyzed, and used to train a model from one data set, this model is then transferred to another data set, and so on: this allows to improve the performance of the model through coordination among the different data sets.

## 8. Requirements of DJL in future networks including IMT-2020

Based on the high-level architectural requirements for the design of the architectural framework for ML in future networks including IMT-2020 described in [ITU-T Y.3172], and the high-level requirements for ML model serving described in [ITU-T Y.3179], the following high-level requirements of DJL in future networks including IMT-2020 are derived.

### 8.1 DJL task management

This clause describes high-level requirements related to DJL task management to enable ML in the future networks including IMT-2020.

REQ-DJL-TASK-001: The architectural framework for DJL is required to support task orchestration, including decomposing tasks into different ML functions according to the requirements of tasks.

NOTE - ML functions include data collection and processing, AI model training and AI model training collaboration.

REQ-DJL-TASK-002: The architectural framework for DJL is required to support task assignment and control, including task decomposition, selection of appropriate ML function nodes to participate in federated learning, task balancing according to the status of resources and the task load of all DJL participants, and task life cycle management.

REQ-DJL-TASK-003: The architectural framework for DJL is required to support task execution status tracking, including monitoring of the execution and completion status of tasks, and evaluation of the quality of the task execution process in real time.

REQ-DJL-TASK-004: The architectural framework for DJL is required to support task scheduling and optimization, including DJL node selection and network parameter configuration.

REQ-DJL-TASK-005: The architectural framework for DJL is required to support the coordination of tasks and the synchronization of intermediate results between DJL participants, and the transmission of model parameters and intermediate results among different participants in order to ensure the efficiency of task execution.

### 8.2 DJL node management

This clause describes high-level requirements related to DJL node management to enable ML in the future networks including IMT-2020. The DJL node registration, DJL node selection and DJL node

collaborative mechanisms are required to be supported by the DJL node management functionalities.

REQ-DJL-NODE-001: The architectural framework for DJL is required to support the running of ML functions on DJL nodes.

REQ-DJL-NODE-002: The architectural framework for DJL is required to support the coordination among DJL nodes, including model parameter aggregation and distribution of aggregation parameters.

NOTE 1 - The task of parameter aggregation can be undertaken by a central DJL node or decentralized DJL nodes.

REQ-DJL-NODE-003: The architectural framework for DJL is required to support the participation and exit of DJL nodes in the task training, in order to ensure the stability of DJL node management.

NOTE 2 - The stability of DJL node management refers to the node management adaptation to changes in the number of DJL participating nodes in order to maintain the continuity of the task training.

REQ-DJL-NODE-004: The architectural framework for DJL is required to support optimization mechanisms to improve training efficiency, such as data compression and asynchronous training.

### 8.3 DJL resource management

This clause describes high-level requirements related to DJL resource management to enable ML in the future networks including IMT-2020.

REQ-DJL-RES-001: The architectural framework for DJL is required to support the selection strategy of DJL nodes according to the computing resource status of DJL nodes and the communication resource status of network.

NOTE 1 - As indicated in NOTE 2 in clause 7.2 of [ITU-T Y.3179], the computing resources may include graphics processing unit (GPU) capabilities, central processing unit (CPU) capabilities and field programmable gate arrays (FPGA) capabilities.

NOTE 2 - As indicated in NOTE 4 in clause 7.2 of [ITU-T Y.3179], the communication resources may include the specific radio bearer (RB) and bandwidth to fulfil the latency and throughput requirements.

### 8.4 DJL model management

This clause describes high-level requirements related to DJL model management to enable ML in the future networks including IMT-2020.

REQ-DJL-MODEL-001: The architectural framework for DJL is required to support DJL model storage.

REQ-DJL-MODEL-002: The architectural framework for DJL is required to support the creation, retrieval, update and deletion of DJL models.

NOTE 1 – For a given application scenario, due to different performance requirements, such as accuracy, size, etc., the model training requirements may be different. Therefore, the DJL architecture needs to have the ability to train matching models based on the performance requirements.

REQ-DJL-MODEL-003: The architectural framework for DJL is required to support DJL model partition and DJL model aggregation.

NOTE 2 – As an example, in some application scenarios where deep neural network models are trained, the training cost of large models is very high. By segmenting the deep neural network model, some DJL nodes are responsible for training the partial model instead of the whole model. After the training of the segmented model is completed, the whole model can be obtained by merging the partial models. This model segmentation strategy reduces the transmission amount of training parameters at a specific time and improves the communication efficiency.

NOTE 3 – Due to the continuous generation of data and the iterative update of requirements, the trained model needs to be iterated to adapt to the latest requirements, so model aggregation is a way

to update the model iteratively. In this case, the weighted aggregation of models is supported according to the performance of models.

REQ-DJL-MODEL-004: The architectural framework for DJL is required to support the secure management of models.

REQ-DJL-MODEL-005: The architectural framework for DJL is required to support the model version control for comparison of models and rollback to previous model versions if needed.

REQ-DJL-MODEL-006: The architectural framework for DJL is required to support the logging of models to track model usage.

### **8.5 DJL data management**

This clause describes high-level requirements related to DJL data management to enable ML in the future networks including IMT-2020.

REQ-DJL-DAT-001: The architectural framework for DJL is required to support the management of data from different data sources.

REQ-DJL-DAT-002: The architectural framework for DJL is required to support the storage of data from multiple data sources.

REQ-DJL-DAT-003: The architectural framework for DJL is required to support the real-time acquisition of distributed data.

REQ-DJL-DAT-004: The architectural framework for DJL is required to support the data real-time backup and retrieval.

### **8.6 DJL interfaces**

This clause describes the high-level requirements related to DJL interfaces to enable ML in the future networks including IMT-2020.

REQ-DJL-INT-001: Compatible and standardized interfaces are required to support coordination among datasets sharing the same features and characteristics for HFL.

REQ-DJL-INT-002: Compatible and standardized interfaces are required for coordinating intermediate results between decentralized data collecting, storage, and processing entities for VFL.

REQ-DJL-INT-003: Compatible and standardized interfaces are required to support model reuse and/or transfer for FTL in future networks.

### **8.7 DJL connectivity**

This clause describes high-level requirements related to DJL connectivity to enable ML in the future networks including IMT-2020.

REQ-DJL-CON-001: The architectural framework for DJL is required to support the data transfer among DJL nodes, and the transmission delay is required to be less than the maximum tolerable delay of each DJL node.

### **8.8 DJL reliability management**

This clause describes high level requirements related to DJL reliability to enable ML in the future networks including IMT-2020.

REQ-DJL-ABN-001: The architectural framework for DJL is required to support the real-time detection of abnormal conditions.

REQ-DJL-ABN-002: The architectural framework for DJL is required to support the recording of information concerning self-correction and self-adjustment based on the abnormal events handling.

### 8.9 DJL security and privacy protection

This clause describes high-level requirements related to DJL security and privacy protection to enable ML in the future networks including IMT-2020.

REQ-DJL-SEC-001: The architectural framework for DJL is required to support the privacy preservation of training data and training models.

NOTE 1 - Privacy preservation refers to avoiding information leakage. The aggregated information from the models provided by DJL clients does not leak the source data information, and attackers cannot infer the participants' information by eavesdropping model parameters during the training or inference stages.

REQ-DJL-SEC-002: The architectural framework for DJL is required to support DJL nodes which can trust each other to avoid false attacks. NOTE 2 – This also allows DJL nodes to receive and send data from/to other components without being blocked.

REQ-DJL-SEC-003: The architectural framework for DJL is required to support data security mechanisms to support data privacy, such as differential privacy and homomorphic encryption.

REQ-DJL-SEC-004: The architectural framework for DJL is required to support the identification of exceptions of training updates from DJL clients, and the deletion of training updates which may cause communication or processing errors.

### 9. Framework of DJL in future networks including IMT-2020

This clause specifies the architectural components which are essential parts of the DJL architectural framework. Integration of such components to a network architecture by interfacing with the ML underlay network, along with the placement of the ML functionalities in such a network, forms the architecture framework of DJL.

#### 9.1. The architectural components for DJL

The architectural components for DJL, as shown in Figure 9-1, comprise five different components: ML Consumer, MLFO Services function, Data Services function, Training Services function and Model repository.

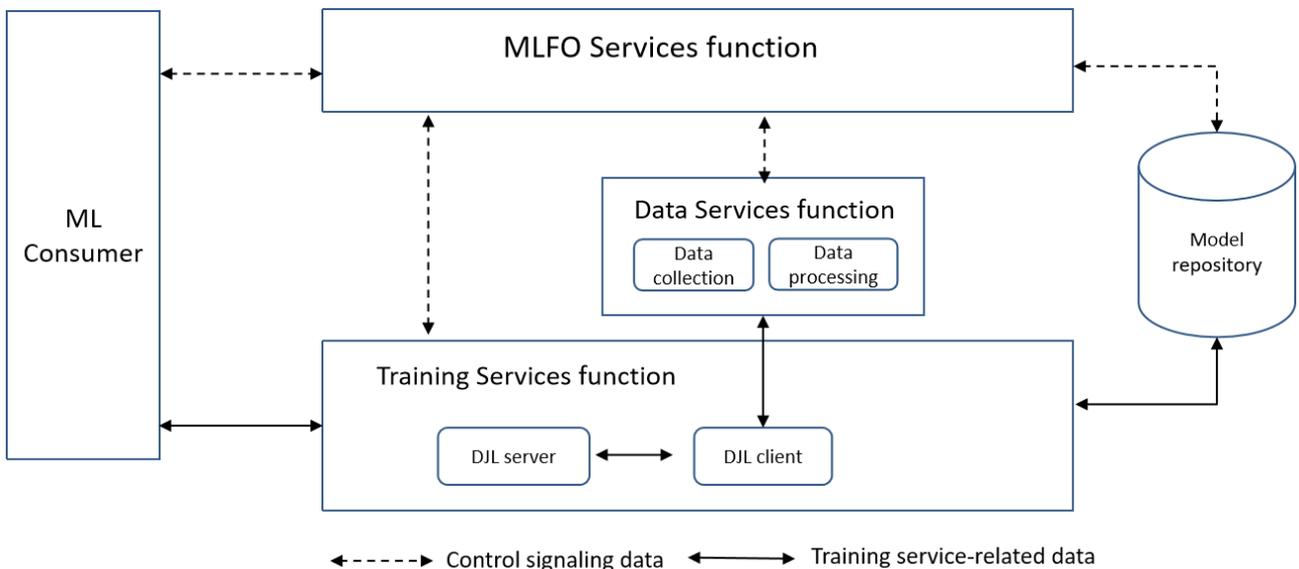


Figure 9-1 Architectural components for DJL

With reference to Figure 9-1, the control signaling data arrows show the paths of control signal interaction between DJL components, and the training service-related data arrows show the path of the training traffic data between DJL components.

[ITU-T Y.3115] specifies a cross-domain collaboration framework for the process of achieving overall network intelligence. The architecture framework of DJL is the enhancement of this cross-domain collaboration framework: it specifies in more detail the MLFO services function, the training services function and the data services function, as well as the interaction between these functions; it also specifies deployment modes of these functions to support DJL. The following sub-clauses detail the different architectural components.

### **9.1.1. Training Services function**

This component is a ML function that consists of specific functionalities for DJL, including DJL server and DJL client functionalities.

The nodes with DJL functionalities can be deployed in a distributed manner, where a DJL client node is a distributed learning execution node, which is responsible for model training based on local data, and a DJL server node is a parameter aggregation node, which is responsible for aggregating model parameters of several DJL client nodes in order to achieve joint learning among multiple DJL client nodes without exposing their private data. This distributed deployment constitutes a typical deployment of the training services function.

NOTE 1 – The training services function component provides training services as defined in [ITU-T Y.3115].

Specifically, the DJL client gets training data from the corresponding data service according to the configuration of the MLFO services function and performs iterative training processes for several rounds. The parameter value in training model is sent to the DJL servers, and the servers aggregate the gradient information from multiple DJL clients according to the aggregation policy to generate new model parameters. Then the new model parameters are sent to the DJL clients, and the DJL clients continue to execute the next round of training process until the training exit criteria are met. Finally, the trained model is sent to the ML consumer and the MLFO services function is notified.

NOTE 2 – The DJL server and the DJL clients are registered to the MLFO services function so that they can be discovered during the orchestration of DJL (see clause 9.1.4).

NOTE 3 – The training services function receives the DJL task instructions from the MLFO services function. The QoS requirement of DJL task may include the accuracy and the size of the generated AI model.

NOTE 4 – Gradient [b-Shome] is an example of the parameters used for iterative optimization in DJL algorithms.

### **9.1.2. Data Services function**

This component is a ML function that consists of specific functionalities for DJL, including functionalities of data collection and data processing, the last one including data cleaning and data normalization. The DJL nodes supporting data collection and/or data processing can be deployed in a distributed manner. The component provides data services to the training services function.

NOTE 1 – The data services function provides authorized consumers with the capabilities of collecting, processing, retrieving and sharing data as defined in [ITU-T Y.3115].

NOTE 2 – The normalization of the data [b-ETSI GS ENI 005] enables the ML models to learn more quickly the optimal parameters for each DJL node. It also reduces complex computational problems associated with floating point number precision.

### 9.1.3. ML Consumer

This component is the consumer of DJL services.

It requests DJL services from the MLFO services function. The training services function performs specific DJL tasks and returns the DJL results to the ML consumer.

The ML consumer may be not only an internal NF of the IMT-2020 network, i.e., an internal ML consumer, but also an external ML consumer, such as a UE, an AF, which requests DJL services from the IMT-2020 network via an exposure interface.

### 9.1.4. MLFO Services function

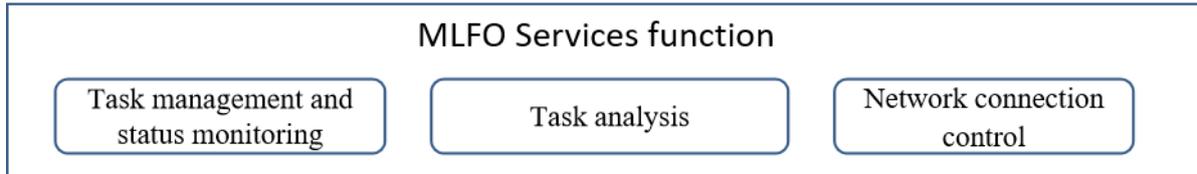


Figure 9-2 Functionalities of the MLFO Services function

This component is responsible for the management and orchestration of the DJL nodes of ML pipelines in the IMT-2020 network [ITU-T Y.3172].

It includes three functionalities, namely task management and status monitoring, task analysis, and network connection control, as shown in Figure 9-2. The task management and status monitoring functionality and the task analysis functionality are jointly responsible for analyzing and orchestrating tasks in the MLFO, and the network connection control functionality is responsible for monitoring, controlling, and optimizing the data transmission connections between DJL nodes. The following are details on the three functionalities:

- The task management and status monitoring functionality is responsible for the information registration and dynamic monitoring of the DJL nodes. The complex information of DJL nodes (e.g., node ID, survival status, resource status, data category, etc.) and DJL nodes' relationships (e.g., different DJL nodes with complementary or similar data categories for the same DJL task, etc.) may be organized and extracted based on the knowledge graph technique [b-Tian].
- The task analysis functionality is responsible for parsing the DJL service requests from the ML consumer into corresponding DJL tasks, providing the analysis of the selection strategy of DJL nodes, and orchestrating the DJL nodes of ML pipelines [ITU-T Y.3172]. The analysis of the DJL node selection strategy is based on the requirements of the DJL tasks and the DJL node information from the task management and status monitoring functionality.
- The network connection control functionality is responsible for the optimization of data transmission connections when the quality of the network connections between the DJL nodes does not meet the QoS requirements of the DJL tasks.

NOTE 1 – [ITU-T Y.3172] defines the MLFO as a logical node with functionalities that manage and orchestrate the nodes of a ML pipeline. The architectural framework of DJL extends the MLFO functionalities as described in [ITU-T Y.3172] in order to support DJL scenarios.

NOTE 2 – The knowledge graph technique is a technique for organizing information in a structured way and explicitly describing complex relationships between entities. In DJL, a knowledge graph may be constructed by using the information of multiple DJL nodes in order to represent the relationships among DJL nodes and make the selection of DJL nodes fast and comprehensive.

**9.1.5. Model Repository**

This component is a repository of ML models, from where ML models can be retrieved for the purpose of ML model serving [ITU-T Y.3179].

**9.2. Architectural framework**

The components for DJL described in clause 9.1 can be integrated into the architectural framework for DJL in future networks including IMT-2020, as shown in Figure 9-3.

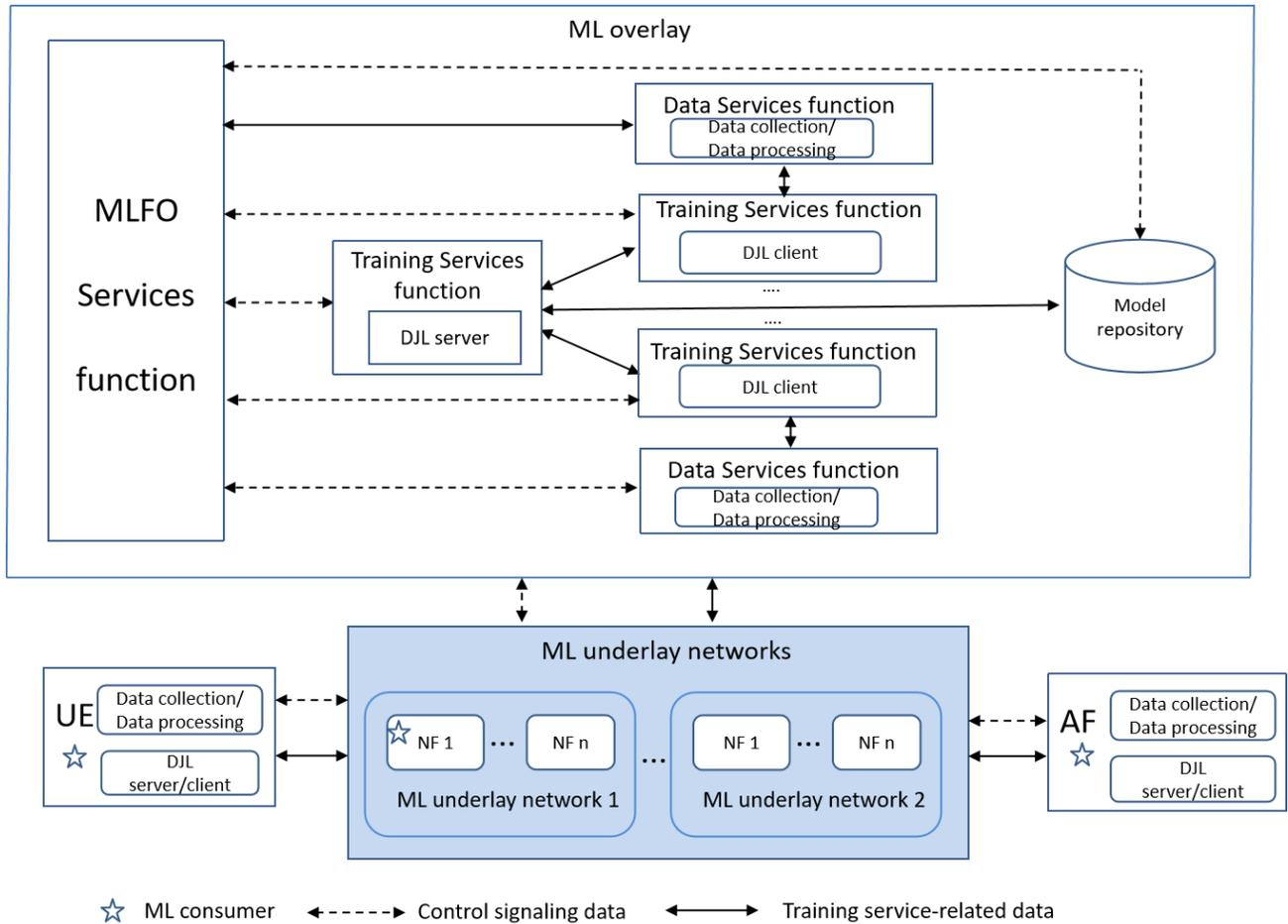


Figure 9-3 Architectural framework for DJL

The integration of the components for DJL by interfacing them with the NFs of the ML underlay network, along with the placement of the ML functionalities in such a network, forms the architecture framework for DJL.

The ML overlay comprises machine learning components for DJL including ML consumer, MLFO services function, data services function, training services function, and model repository, as described in clause 9.1.

The DJL components (within the ML overlay) interact with each other to handle control signaling data required for DJL tasks and to handle training service-related data required for DJL tasks.

The DJL components (within the ML overlay) interact with the ML underlay network to handle control signaling data required for DJL service requests and notifications, as well as to handle training service-related data required for DJL tasks when collecting network and external data.

The DJL nodes with the functionalities of the training services function and the data services function can be deployed in a distributed manner. These DJL nodes can not only be deployed on a machine learning overlay, but also on UEs and AFs.

NOTE 1 – Figure 9-3 shows a distributed deployment of the training services function component (the DJL server and DJL client functionalities are typically distributed).

The ML underlay networks as described in [ITU-T Y.3172] are a set of telecommunication networks, and their related network functions, which interface with corresponding machine learning overlays.

NOTE 2 – The two distinct ML underlay networks (ML underlay network 1 and ML underlay network 2) aim to illustrate different types of technology-specific ML underlay networks, such as an IMT-2020 network [ITU-T Y.3104] and an IP network [ITU-T Q.1742.3].

### **9.3. Deployment options of the DJL components**

#### **9.3.1. DJL nodes deployed on a machine learning overlay**

The DJL nodes of MLFO, DJL server, DJL client, data collection and data processing are deployed on a machine learning overlay in the IMT-2020 network. This option is applied for scenarios of DJL with the use of network data, which are described in clauses 7.1.1.

The DJL nodes deployed in a distributed manner on a ML overlay interact and exchange training service-related data with other DJL nodes in the ML underlay network.

#### **9.3.2. DJL nodes deployed on UEs and AFs**

The DJL nodes of DJL server, DJL client, and data collection and data processing are deployed on UEs and AFs. This option is applied for scenarios of DJL with the use of network data and external data, which are described in clauses 7.1.2.

The DJL nodes deployed on UEs interact with other DJL nodes for control signaling data in the control plane and exchange training service-related data with other DJL nodes in the user plane in the IMT-2020 network.

The DJL nodes deployed on AFs interact with other DJL nodes for control signaling data through the exposure interface of the ML underlay network, and exchange training service-related data with other DJL nodes in the ML underlay network.

### **9.4. DJL data transmission optimization**

As described in the requirement REQ-DJL-CON-001, the quality assurance of data transmission between DJL nodes is important for the performance of DJL. NOTE - During the running of DJL, a large amount of training service-related data is transmitted among DJL server, DJL clients and data services function. Therefore, high-quality transmission is required, otherwise, the performance of DJL may be affected.

In order to achieve the optimization of data transmission between DJL nodes, the MLFO interacts with the ML underlay network to establish optimal data transmission connections in the ML underlay network. The following sub-clauses describe how data transmission optimization is achieved.

#### **9.4.1. DJL data transmission optimization based on network control in the ML underlay network**

When DJL nodes are deployed on a machine learning overlay or AFs, data transmission between the DJL nodes is carried on the ML underlay network. The dynamic optimization of the data transmission connections between DJL nodes can be achieved by the network control functions in the ML underlay network, upon request by the MLFO.

NOTE - Let's take the IP network as an example of ML underlay network: an IP network includes two relevant components from the data transmission viewpoint, SDN controller and Router. The SDN controller is responsible for the routing policy in the IP network, the Router executes the routing policy. They cooperate with each other to realize the routing and optimization of IP-based packet data transmission. When the quality of the data transmission between the DJL nodes does not meet the service requirements, the MLFO may interact with the IP network to request the optimization of the data transmission connections, so that the performance of the DJL may be improved.

**9.4.2. DJL data transmission optimization based on policy control in the IMT-2020 network**

When DJL nodes are deployed on UEs, the dynamic optimization of the data transmission connections between DJL nodes can be achieved by PCF (Policy control function) [ITU-T Y.3102], which is responsible for the control and management of policy rules for QoS enforcement. When the quality of the data transmission between the DJL nodes does not meet the service requirements, the MLFO sends a data transmission connection optimization request to the PCF, and the PCF configures policy rules, including rules for QoS enforcement and traffic routing of the UE's IP flow [ITU-T Y.3102] according to the data transmission connection optimization requirements in order to achieve the data transmission optimization.

**10. Flow diagram for DJL**

Figure 10.1 illustrates the flow diagram for DJL.

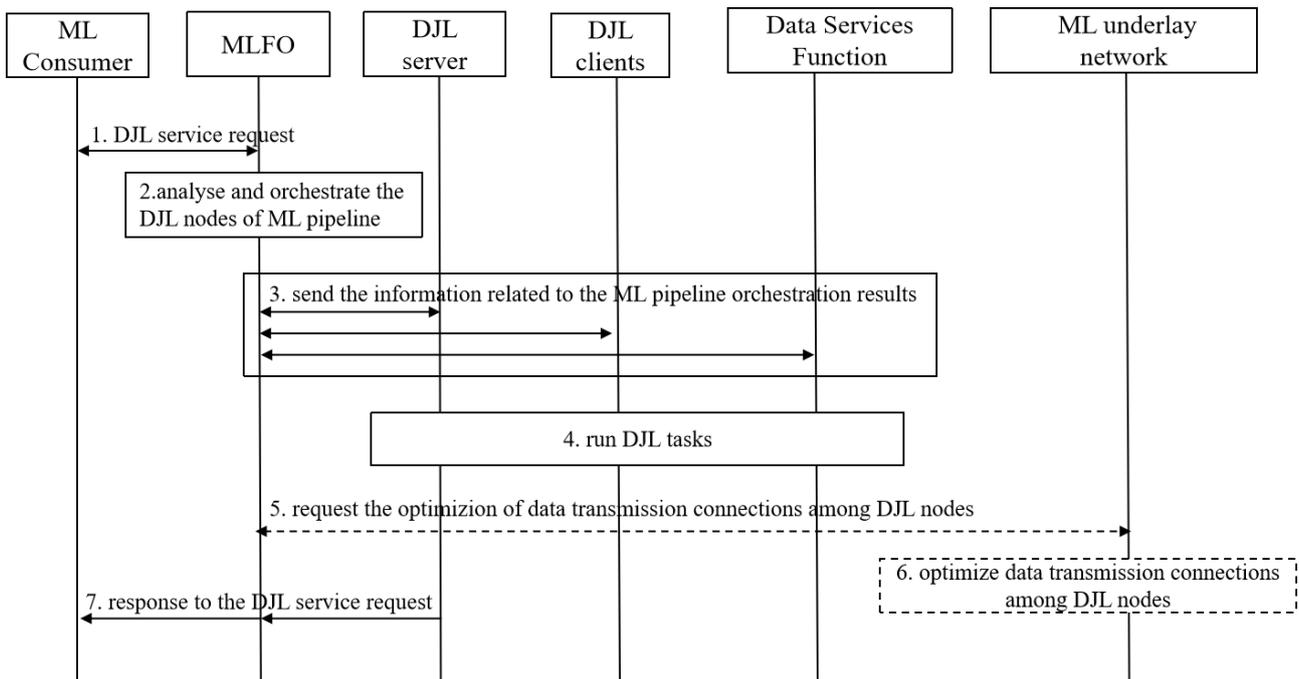


Figure 10-1 Flow diagram for DJL

1. The ML consumer sends a DJL service request to the MLFO.
2. Upon reception of the DJL service request, the MLFO analyses the strategy for DJL node selection and orchestrates the DJL nodes of the ML pipeline.
3. The MLFO sends the information related to the ML pipeline orchestration results to the selected DJL nodes including DJL server, DJL clients and Data Services Function.

4. The selected DJL nodes collaborate with each other to run DJL tasks, including data collection, data processing, model training, model parameter aggregation.

5. When the MLFO detects that the data transmission quality among the DJL nodes is lower than the set threshold, the data transmission optimization process is triggered. The MLFO requests the network control functions to optimize the data transmission connections among the DJL nodes in the ML underlay network

6. The optimization of the data transmission connections among the DJL nodes is performed in the ML underlay network.

NOTE - Step 5 and step 6 can be triggered on demand throughout the running of the DJL tasks.

7. The response to the DJL service request is notified to the ML consumer after the DJL tasks are completed.

## 11. Security considerations

The security and privacy related requirements specified in [ITU-T Y.3172] are applicable to this Recommendation.

High-level requirements related to DJL security and privacy protection are addressed in clause 8.9.

When the IMT-2020 network exposes its capabilities to third parties, including by exchanging DJL control signaling data, authentication and authorization of third parties to access the exposed network capabilities is required to be performed.

## Bibliography

[b-3GPP TS 22.186] Recommendation 3GPP TS 22.186 (2021), Enhancement of 3GPP support for V2X scenarios.

[b-ETSI GS ENI 005] ETSI GS ENI 005 V4.10.1 (2023-08) “Experiential Networked Intelligence (ENI); System Architecture”

[b-Shome] Shome D, Waqar O, Khan W U. Federated learning and next generation wireless communications: A survey on bidirectional relationship[J]. 2021.

[b-Tian] L. Tian, X. Zhou, Y. P. Wu, W. T. Zhou, J. H. Zhang, & T. S. Zhang. Knowledge graph and knowledge reasoning: a systematic review. Journal of Electronic Technology (002), 020. 2022

[b-Yang] Yang Q, Liu Y, Cheng Y, et al. Federated learning[J]. Synthesis Lectures on Artificial Intelligence and Machine Learning, 2019, 13(3): 1-207.

---