



INTERNATIONAL TELECOMMUNICATION UNION

**TELECOMMUNICATION
STANDARDIZATION SECTOR**

STUDY PERIOD 2017-2020

SG13-TD223/PLEN

STUDY GROUP 13

Original: English

Question(s): 20/13

Geneva, 14-25 October 2019

TD

Source: Editors

Title: Draft new Recommendation ITU-T Y.3174 (formerly Y.ML-IMT2020 -Data-Handling): " Framework for data handling to enable machine learning in future networks including IMT-2020" – for consent

Purpose: Proposal

Contact: Qi Sun
China Mobile
P.R. China
E-mail: sunqiyjy@chinamobile.com

Contact: Yaxing Qiu
China Mobile
P.R. China
E-mail: qiuyaxing@chinamobile.com

Contact: Liya Yuan
ZTE
P.R. China
E-mail: yuan.liya@zte.com.cn

Contact: Salih Ergüt
Turkcell
Turkey
Tel: +90-533-210-9958
E-mail: salih.ergut@turkcell.com.tr

Contact: Xin Guo
Lenovo
P.R. China
E-mail: guoxin9@lenovo.com

Contact: Yameng Li
China Unicom
P.R. China
E-mail : liy710@chinaunicom.cn

Keywords: Data handling; Machine Learning; training, IMT-2020; requirements; architecture

Abstract: This is the output draft of Recommendation ITU-T Y.ML-IMT2020-Data-Handling developed at the Q20/13 Geneva meeting, October 2019.

The following text is the update of draft Recommendation ITU-T Y.ML-IMT2020- Data-Handling; it is put forward for consent.

Draft new Recommendation ITU-T Y.3174 (formerly Y. ML-IMT2020-Data-Handling)
Framework for data handling to enable machine learning in future networks
including IMT-2020

Summary

A framework for data handling to enable machine learning in future networks including IMT-2020 is described in this Recommendation. The requirements for data collection and processing mechanisms in various usage scenarios for machine learning in future networks including IMT-2020 are identified along with the requirements for applying machine learning output in the machine learning underlay network. Based on this, a generic framework for data handling and examples of its realization on specific underlying networks are described.

Keywords

Data handling, Machine Learning, training, IMT-2020, requirements, architecture

CONTENTS

	Page	
1	Scope	6
2	References.....	6
3	Definitions.....	7
3.1	Terms defined elsewhere	7
3.2	Terms defined in this Recommendation	8
4	Abbreviations and acronyms.....	9
5	Conventions	11
6	Introduction.....	11
7	High-level requirements for data handling	13
7.1	High Level Requirements	13
7.1.1	Storage requirements in ML data collection	13
7.1.2	ML-data-collection-level	14
7.1.3	ML-data-collection-timing	15
7.1.4	ML-data-collection-statistics	15
7.1.5	ML-data-collection-data model	15
7.1.6	ML-data-collection-specification	17
7.2	High Level Requirements for ML data output	17
7.2.1	ML-data-output-levels	17
7.2.2	ML-output-data model	18
7.2.3	ML-output-policy	19
7.2.4	ML-output-timing.....	19
7.3	High Level Requirements for ML processing	20
7.3.1	ML-processing-models.....	20

7.3.2	ML-processing-data.....	20
7.3.3	ML-processing-levels	20
7.3.4	ML-processing-KPI.....	20
8	High level framework for data handling.....	21
8.1	Framework components.....	21
8.1.1	Reused Components defined in [ITU-T Y. 3172].....	22
8.1.2	New defined data handling framework components.....	23
8.1.2.1	ML Metadata store	23
8.1.2.2	API-g	23
8.1.2.3	API-s	24
8.1.2.4	ML Data Broker Control Plane (DBr-CP)	24
8.1.2.5	ML Data Broker User Plane (DBr-UP)	25
8.1.2.6	ML Data Base (DB)	25
8.1.3	Data handling framework supporting components.....	25
8.1.3.1	ML model repository	25
	This supporting component is a repository of machine learning models.	26
8.2	High-level architecture	26
8.3	Sequence diagrams for ML data handling	28
8.3.1	Instantiation of data handling framework.....	28
8.3.2	Addition of new SRC in data handling framework.....	31
8.3.3	Data model does not exist in the ML metadata store	35
9	Security considerations	37
	Appendix I.....	38
	Appendix II	40
	Bibliography.....	44

Draft new Recommendation ITU-T Y.3174 (formerly Y. ML-IMT2020-Data-Handling)
Framework for data handling to enable machine learning in future networks
including IMT-2020

1 Scope

This Recommendation provides the framework of data handling to enable machine learning in future networks including IMT-2020.

The scope of this Recommendation includes:

- Background and motivations
- High level requirements of data handling and data models
- Framework for data handling to enable machine learning in future networks including IMT-2020
- Guidelines and examples for usage of the framework in future networks including IMT-2020

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this deliverable. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this deliverable 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 deliverable does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Y.3111] ITU T Recommendation Y.3111 (2017), “IMT-2020 network management and orchestration framework”

[ITU-T Y.3172] ITU-T Recommendation Y.3172 (2019), “Architectural framework for machine learning in future networks including IMT-2020”

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 - Definition adapted from [b-ETSI GR ENI 004].

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

3.1.2 machine learning 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.

NOTE – The nodes of a machine learning pipeline are entities that are managed in a standard manner and can be hosted in a variety of network functions [b-ITU-T Y.3100].

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

3.1.4 machine learning overlay [ITU-T Y.3172]: A loosely coupled deployment model of machine learning functionalities whose integration and management with network functions, are standardized.

NOTE – A machine learning overlay aims to minimize interdependencies between machine learning functionalities and network functions using standard interfaces, allowing for parallel evolution of functionalities of the two.

3.1.5 machine learning sandbox [ITU-T Y.3172]: An environment in which machine learning models can be trained, tested and their effects on the network evaluated.

NOTE – A machine learning sandbox is designed to prevent a machine learning application from affecting the network, or to restrict the usage of certain machine learning functionalities.

3.1.6 machine learning underlay network [ITU-T Y.3172]: A telecommunication network and its related network functions which interface with corresponding machine learning overlays.

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

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 machine learning data model: The format which describes the data used for data handling in machine learning (ML) applications.

NOTE 1 - The format may specify the data exchanged between ML overlay and ML underlay network.

NOTE 2 – This includes the data structures as well as semantic description used while collecting data from the ML underlay network and while applying ML output from ML overlay to the ML underlay network.

3.2.2 machine learning metadata store: A component which stores the machine learning data models with the corresponding application programming interfaces.

3.2.3 machine learning data broker: A component which maps the machine learning data model (for machine learning data collection and machine learning data output) from the machine learning underlay network to the source and sink nodes of the machine learning pipeline.

3.2.4 machine learning data base: A component which stores data related to machine learning in the machine learning overlay.

3.2.5 application programming interface-generic: A generic set of application programming interfaces for a machine learning data model to be used in machine learning overlay and stored in the machine learning metadata store.

3.2.6 application programming interface-specific: A specific set of application programming interfaces for a machine learning data model to be used by machine learning underlay networks.

NOTE – Application programming interface-specific is used for specific operations on the data which are in the machine learning underlay network (e.g. data collection and configuration management).

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AF Application Function

AF-s Simulated Application Function

AN Access Network

API Application Programmer Interface

BDAP Big Data Application Provider

BDIP Big Data Infrastructure Provider

BDSP Big Data Service Provider

BDSU Big Data Service User

BS Base Station

BTS Base Transceiver Station

CN Core Network

CP Control Plane

CSC Cloud Service Customer

CSN Cloud Service partner

CSP Cloud Service Provider

CU Central Unit

DB Data Base

DM Data Model

DNN Deep Neural Network

DP Data Provider

DU Distributed Unit

E2E	End-to-End
KPI	Key Performance Indicator
MEC	Multi-access Edge Computing
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
NF	Network Function
NF-s	Simulated Network Function
NFV	Network Functions Virtualisation
NFVO	Network Functions Virtualisation Orchestrator
NMS	Network Management Subsystem
PDCP	Packet Data Convergence Protocol
QoS	Quality of Service
RAN	Radio Access Network
RCA	Root Cause Analysis
SBA	Service-Based Architecture
SDO	Standards Development Organization
SDNC	Software-Defined Networking Controller
SIM	Simulator
SO	Service Orchestrator
SRC	Source
UE	User Equipment
UP	User Plane
VIM	Virtualised Infrastructure Manager

VNF Virtualised Network Function

VNFM Virtualised Network Function Manager

5 Conventions

In this Recommendation, requirements are classified as follows:

- The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.
- The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, such requirements need not be present to claim conformance.
- The keywords "can optionally" and "may" indicate an optional requirement which is permissible, without implying any sense of being recommended. These terms are not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the NOP/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

6 Introduction

Machine learning is a significant trend in the industry. There is a broad agreement about the remarkable potential of ML to lead innovation, boost commerce, and drive progress, among leaders in the industry, academia, and government. IMT-2020 network, featured by diverse services, e.g., mobile Internet, Internet of Things, cloud computing and other types of communication, will lead to the growth of data traffic and big data in the network. ML introduces new ways of solving the problems in the future networks. [b-ITU-T Y.ML-IMT2020-Use-Cases] describes several use cases where ML provides innovative and efficient solutions to problems in the future networks including IMT-2020. A high-level architecture framework to enable ML in future networks was described in [ITU-T Y.3172].

The following challenges are addressed in this document:

- A. Various components in the network produce data with differing characteristics. This diversity in network data sources forms one of the biggest challenges to implement the architecture framework defined in [ITU-T Y.3172].

- B. The future networks are expected to be more and more flexible and agile, and this may increasingly complicate the configurations of these networks. As examples, the introduction of flexible air interface, service based architecture (SBA) [b-ETSI 129 500] and central unit (CU)/distributed unit (DU) disaggregated architecture into IMT-2020 networks enables agile deployments with richer configuration options. The increased flexibility and agility make challenging to enable ML in future networks including IMT-2020.
- C. Evolution of network techniques has resulted in evolving sources of data, applicable network configuration parameters and policies. Adapting to the changes in future networks while preserving the quality of data needed for ML applications, is a challenge.

In this Recommendation, a data handling framework for enabling ML in future networks including IMT-2020 is provided to address these challenges.

NOTE 1- See clause 8.2.2 for an explanation on how these challenges are addressed using the data handling framework.

Building on the data handling reference points defined in [ITU-T Y.3172], the data handling framework for ML includes ML data collection, ML data processing and ML data output.

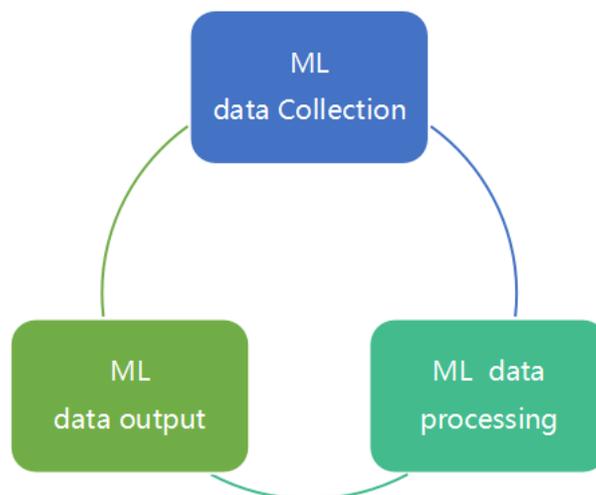


Figure 6-1 General concept of data handling for ML

By analysing various use cases discussed in [b-ITU-T Y.ML-IMT2020-Use-Cases], this Recommendation identifies a set of ML data collection requirements, ML data output requirements and ML data processing requirements. Based on these requirements, a framework for data handling

is proposed for future networks including IMT-2020. The framework aims to solve the challenges listed above and examples for realization of such a data handling framework for a specific underlay are described.

NOTE 2- Implementation of the data handling framework may use existing solutions enabled by cloud computing based big data capabilities [b-ITU-T Y.3600]. Appendix II describes such implementation option.

7 High-level requirements for data handling

By analysing various use cases discussed in [b-ITU-T Y.ML-IMT2020-Use-Cases], the following high level requirements for data handling are derived.

7.1 High Level Requirements

7.1.1 Storage requirements in ML data collection

REQ-ML-DH-001: ML data collection is recommended to provide data storage and management for collected data.

REQ-ML-DH-002: ML data collection is required to support time synchronization of data from various SRCs [ITU-T Y.3172] and output to various SINKs [ITU-T Y.3172].

NOTE 1- The enablement of time synchronization between SRCs and SINKs depends on the ML applications.

REQ-ML-DH-003: It is recommended that, in case of applications where time synchronization is enabled between SRCs and SINKs, time synchronization is maintained in a persistent manner for data which is stored with timestamp in database.

REQ-ML-DH-004: ML data collection is required to support transformation of data while passing data between different ML pipeline nodes.

NOTE 2- Examples of transformation include the mapping of data format between different data models used in the various ML pipeline nodes and the preprocessing of data.

REQ-ML-DH-005: ML data collection is required to be able to define data retention policies at granular level, in accordance with regulations and privacy laws.

REQ-ML-DH-006: ML data collection is required to support run-time queries about data deletion or

archiving.

NOTE 3- Operator may query which data is deleted and the reason for deletion, e.g., data is no more relevant to the ML application.

REQ-ML-DH-007: ML data collection is required to perform time sensitivity analysis of stored data.

NOTE 4– Time sensitivity analysis may indicate, e.g., whether the stored data is still valid after a period of time.

REQ-ML-DH-008: ML data collection is required to perform location sensitivity analysis of stored data.

NOTE 5- Location sensitivity analysis may indicate, e.g., whether the stored data is valid to be used for a ML application with consideration of geographic locations.

REQ-ML-DH-009: ML data collection is recommended to optimize deletion or archive of data.

NOTE 6- This may be based on factors such as predictability of the data, relativity of the data, priority service data preservation, and requirements of ML data models.

REQ-ML-DH-010: ML data collection is required to enable use of archived data, information or context at different levels of ML pipeline.

REQ-ML-DH-011: ML data collection is required to support the ability to select the type of storage for data.

NOTE 7- The type of storage for data includes distributed and centralized storage.

7.1.2 ML-data-collection-level

REQ-ML-DH-012: ML data collection is recommended to be done at various levels in the network.

NOTE 1 - e.g., access network (medium access control (MAC) layer data like MAC data rate), packet data convergence protocol (PDCP) layer data (e.g. PDCP buffer size, base station (BS) load, service type), network management subsystem (NMS) and edge networks.

NOTE 2 - The data collection may be automated, in order to maximize the integration of ML pipelines and network elements.

REQ-ML-DH-013: ML data collection is recommended to enable application programming interfaces (API) for collecting data for ML and applying ML data output over the various levels in the network.

REQ-ML-DH-014: ML data collection is required to support collection of data from the operator hosted network functions or from outside the operator hosted network functions.

NOTE 3- The data sourced outside the operator-hosted network functions may include channel measurement data and environmental data obtained from probes and sensors.

NOTE 4- Environmental data may include: position estimation e.g., global positioning system (GPS), external network detection information and information obtained from users, e.g. propagation environment; three dimensional (3D) maps, transmitter and receiver location; reflectors and scatters; material EM properties; surface roughness; and propagation mechanisms.

NOTE 5- Further examples of data sourced outside the operator-hosted network may be user equipment (UE) and base station (BS) location and location/mobility prediction, e.g. GPS or channel information inferred distance and angular information, geo-tag traffic data with additional context information, e.g., cell identifier (Cell ID), number of antennas, spatial information.

7.1.3 ML-data-collection-timing

REQ-ML-DH-015: ML data collection is required to support diverse frequencies of data collection including continuous data collection.

REQ-ML-DH-016: ML data collection is required to support real-time collection of large volumes of traffic.

7.1.4 ML-data-collection-statistics

REQ-ML-DH-017: ML data collection is required to support collection of real-time statistics.

NOTE - Examples of real-time statistics include network performance data, current number of UEs and traffic speed of the UEs, radio bearer (RB) utilization of current cell and neighbor cell, the number of RRC connections requests in current cell and neighbor cell, physical and virtual resource status.

7.1.5 ML-data-collection-data model

REQ-ML-DH-018: ML data collection is required to support a variety of structures and characteristics for the data.

NOTE 1- Examples of structures include geographic location, events, alarms and log files.

NOTE 2- The structure of data may depend on the type of ML underlay networks and ML applications.

REQ-ML-DH-019: ML data collection is required to support dynamic addition of SRCs associated with new data models.

REQ-ML-DH-020: ML data collection is required to support simulated data.

REQ-ML-DH-021: ML data collection is required to be able to reuse existing interfaces to input data from network functions (NFs) in the underlying networks.

REQ-ML-DH-022: ML data collection is required to support the capability to use application-specific data model.

NOTE 3- An application function (AF) may support specific data models for output of data and input of configurations into the AF. The AF and the specific data model may be specified in the ML Intent.

REQ-ML-DH-023: ML data collection is recommended to support the capability to use stored knowledge [b-ITU-T Y.ML-IMT2020-Use-Cases] derived from real and simulated environments.

NOTE 4- Data may be generated by the simulators in ML sandbox for training of ML models. This can generate knowledge which can be used in real network deployment of the ML pipeline.

REQ-ML-DH-024: ML data collection is recommended to support collection of resource status and topology of the network.

NOTE 5-This capability may be used for the correlation with alarms and events.

REQ-ML-DH-025: ML data collection is recommended to record the metadata related to each session of data collection.

NOTE 6- Examples of such metadata are location of measurement, system configuration parameters like antennas parameters, multiple input multiple output (MIMO) mode, frequency band, transmit power and bandwidth.

REQ-ML-DH-026: ML data collection is required to support performance counters and KPIs collected from the network functions at the radio access, transport, and core network levels.

NOTE 7- This may include data from network slices at various levels. Network state data may include cell id, number of users per cell, number of handovers per cell, etc.

REQ-ML-DH-027: ML data collection is required to include the ability to collect user plane data along with relevant metadata.

NOTE 8- Examples of metadata can be protocol fields, 5-tuple (IP source address, IP destination address, source port, destination port, and protocol) and network topology.

7.1.6 ML-data-collection-specification

REQ-ML-DH-028: ML data collection is required to have the ability to be configured by network operator policies.

REQ-ML-DH-029: ML data collection is required to be controlled by the specification of ML applications.

NOTE 1 – The specification of ML applications may include: a) latency criteria bounding the ML data collection, which influence the positioning of the related data handling components; (b) the overheads of training, e.g. minimum sample size, which influence the size of data collected for training.

REQ-ML-DH-030: ML data collection is required to be done securely, based on the ML application's requirements.

REQ-ML-DH-031: ML data collection is required to comply with all regulations while optimizing the size of collected data.

NOTE 2– Examples of optimization methods for the size of collected data are compression and quantization.

REQ-ML-DH-032: ML data collection is required to understand and report the overhead of training for ML applications.

NOTE 3- The report may include the estimation of latency and resource requirements to transfer, receive and store information and training data across the network.

7.2 High Level Requirements for ML data output

7.2.1 ML-data-output-levels

REQ-ML-DH-033: ML data output is recommended to have the ability to be applied at the different phases of the network design, operation and management.

REQ-ML-DH-034: ML data output is required to have the ability to be applied to network functions at any levels.

NOTE 1- ML data output may be applied to network functions in, e.g., multi-access edge computing (MEC), access network (AN), core network (CN), gNB, NMS, central and edge servers, including their adaptive real time configurations.

NOTE 2- ML data output may be used for, e.g., proactive resource allocation, optimized handovers, predictive caching, advanced energy saving schemes, transmission control protocol (TCP) window decision model, TCP packet transmission rate, and scheduling of UEs among different cells.

NOTE 3- ML data output may be used for automatic recovery actions.

REQ-ML-DH-035: ML data output may be applied using a variety of structures and characteristics of parameters in the ML underlay networks.

REQ-ML-DH-036: ML data output is required to reuse existing standard protocols while applying the output across multiple levels and to different underlay technologies, wherever possible.

REQ-ML-DH-037: ML data output is required to support abstraction of network configuration where needed and hence enable it to be applied at various granularities of network levels.

NOTE 4- Configuration may be done at AN level or CU or DU level. Correspondingly the parameters and interfaces need to be adapted and the right level of abstraction and interface needs to be applied.

REQ-ML-DH-038: ML data output is required to support efficient management of network resources.

NOTE 5- ML data output may interface with, e.g., the cloud orchestrator in network slice resource management [ITU-T Y.3111].

7.2.2 ML-output-data model

REQ-ML-DH-039: ML data output is required to be able to reuse existing interfaces to configure the network functions (NFs) in the underlying networks.

REQ-ML-DH-040: ML data output is recommended to support adaptive real-time network configuration.

NOTE 1- ML data output may include (but not limited to) information for proactive resource allocation, optimized handovers, predictive caching, and advanced energy saving schemes.

REQ-ML-DH-041: ML data output is required to interface with the NMS for network configuration management.

NOTE 2- ML data output may include, e.g., indication to the NMS whether to perform cell splitting/merging and choose a related set of configuration parameters, in order to balance the UEs connected to the radio access network (RAN) managed by the NMS.

REQ-ML-DH-042: ML data output is required to support dynamic addition of SINKs with new schemas for configuration.

NOTE 3- The schemas are unknown at design time and revealed at run-time.

NOTE 4- As an example, ML output may lead to tagging the user-plane packets.

REQ-ML-DH-043: ML output is recommended to be used for optimizing the performance of the network.

NOTE 5- As an example, optimizing the performance may include scaling-in or scaling-out of network resources.

7.2.3 ML-output-policy

REQ-ML-DH-044: ML data output is required to be controlled by network operator policies.

REQ-ML-DH-045: ML data output is required to be done securely.

REQ-ML-DH-046: ML data output is required to maintain its integrity and consistency when applied to NFs in underlying networks and be not affected by the methods used in managing ML data collection, ML processing and ML output.

7.2.4 ML-output-timing

REQ-ML-DH-047: ML data output is required to be bounded by latency criteria which are specified by ML applications.

REQ-ML-DH-048: ML data output is required to support real-time configuration in the underlying network.

NOTE 1- ML data output may interface with NFs that perform real time processing, e.g., the BS scheduler.

NOTE 2- Real time configuration depends on the capabilities exposed by the NFs in the ML underlay.

7.3 High Level Requirements for ML processing

7.3.1 ML-processing-models

REQ-ML-DH-049: ML processing is required to use a data model which is general and agnostic to the underlying technology to manipulate data used in different network management and operation systems.

7.3.2 ML-processing-data

REQ-ML-DH-050: ML processing is required to support the analysis and learning of a large volume of data.

REQ-ML-DH-051: ML processing is recommended to use data sets from simulated or live ML underlay networks for training.

REQ-ML-DH-052: ML processing is required to be done securely.

7.3.3 ML-processing-levels

REQ-ML-DH-053: ML processing may be hosted at various levels in the network.

7.3.4 ML-processing-KPI

REQ-ML-DH-054: ML processing is required to optimize size of collected data.

REQ-ML-DH-055: ML data processing is required to be bounded by the latency criteria which is specified by ML applications.

REQ-ML-DH-056: ML data processing is required to understand and report the overhead of training for ML applications.

8 High level framework for data handling

A high level framework of data handling which supports the requirements defined in clause 7 is described in this clause. This includes the description of the framework components and sequence diagrams for data handling operations.

Building on high-level architecture framework [ITU-T Y.3172], the data handling framework enables the use of data models for ML functionalities and corresponding APIs, which are independent of the (technology-specific) underlay networks, in the following ways:

- a) Data models published by standards development organizations (SDO) or industry bodies can be used where available. These data models can be referred in the ML intent and can be downloaded from repositories, e.g. ML model repositories which host ML models trained using data which follows these data models. Generally, by reusing published data models where available, the need for defining new data models for ML is reduced and above all, allows training of ML models based on such data models, before they are deployed in the ML pipeline. See figure 8.3 below.
- b) Data models specific to implementations may be referred in the ML intent. These data models can be downloaded from corresponding repositories, e.g. ML model repositories which host ML models. This option gives network operators the flexibility to use such unpublished data models to train ML models. See figure 8.5 below.

The identified data model and corresponding APIs have to be mapped to the data used in the technology-specific underlay networks. Appendix I gives an example of applying the data handling framework on technology-specific underlay networks.

8.1 Framework components and interfaces

This clause describes the high level architecture components of the data handling framework for enabling the machine learning in future networks including IMT-2020. Integration of such components to a ML underlay network by interfacing with NFs, along with the placement of the ML functionalities, forms the data handling framework.

Figure 8-1 shows the high-level architecture components of the data handling framework.

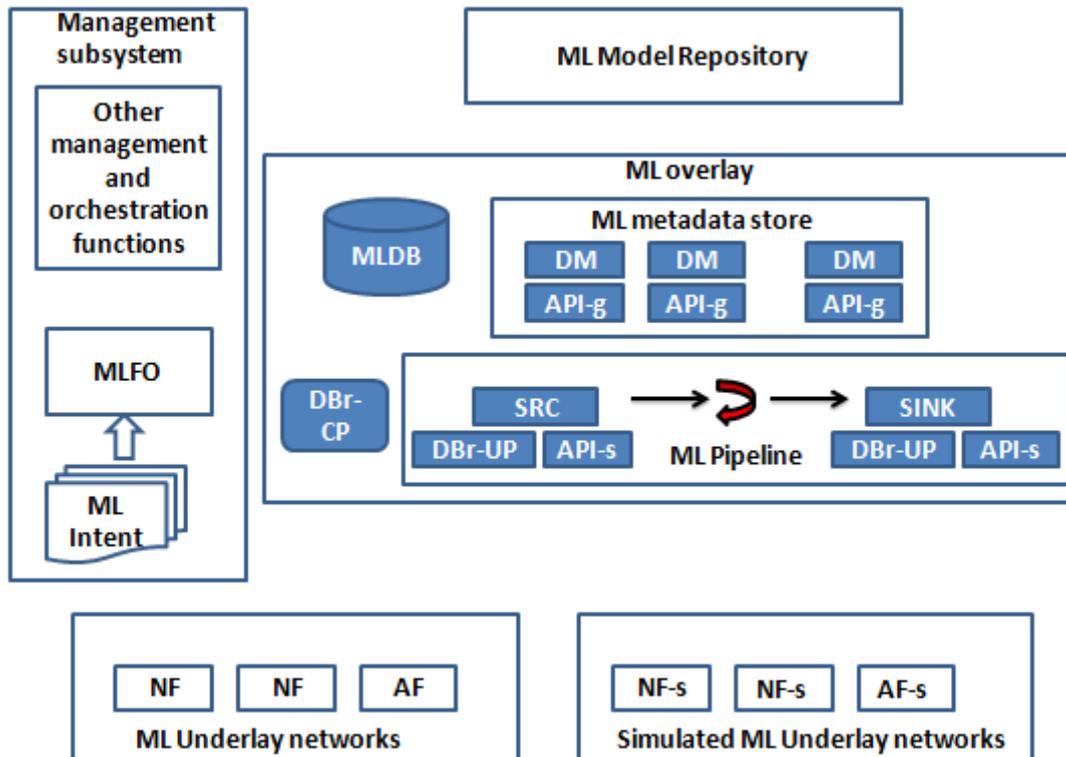


Figure 8.1 High-level architecture components of the data handling framework

8.1.1 Reused components defined in [ITU-T Y. 3172]

The following high-level architectural components defined in [ITU-T Y. 3172] are reused in the framework for data handling:

1) **ML sandbox:** an environment in which machine learning models can be trained, tested and their effects on the network evaluated.

NOTE - The simulated ML underlay networks is a part of ML sandbox.

2) **ML pipeline:** a machine learning pipeline is a set of logical nodes, each with specific functionalities, that can be combined to form a machine learning application in a telecommunication network. In particular, the ML pipeline includes the following nodes:

- **SRC:** this node is the source of data that can be used as input to the ML pipeline.
- **SINK:** this node is the target of the ML output, on which it takes action.

3) **MLFO**: a logical node with functionalities that manage and orchestrate the nodes in a machine learning pipeline.

8.1.2 Newly defined data handling framework components and interfaces

8.1.2.1 ML metadata store

ML metadata store is a component which stores the data models usable in the network for ML applications. Data model specifications stored in the ML metadata store may have companion API specifications denoted as API-g as shown in Figure 8.1.

The purpose of such data models includes application management, resource management and network management. Many vendors implement such data models in their products, sometimes with extensions and specific customizations. While specifying the ML applications, the use case author can re-use any such standard data model, but the following challenges remain:

- a) Vendor implementations of such data models in the ML underlay may differ.
- b) A given ML application may need access to data from multiple levels of ML underlay networks (e.g. core network, access network, management plane). It is also possible that a given ML application may need access to data from varied ML underlays. Thus, a unified view of data model is needed for machine learning use cases.

The data models specified for management purposes may not be sufficient for ML applications, nor may they be applicable as it is to ML applications. In such cases, additional effort may be required to enable and map data models from ML underlays to the data models. Hence the data models and corresponding API-s (API-g) to be used for the ML applications are stored in the metadata store.

NOTE - The data models stored in ML metadata store are used as abstraction for data used across domains and across ML underlay networks for enabling ML applications. Specifications of data models where available (e.g. from SDOs such as 3GPP, O-RAN, TMF), may be imported and reused from ML model repositories.

8.1.2.2 API-g

API-g specifies the API for a machine learning data model to be used in machine learning overlay. By defining it, ML processing can be implemented without depending on the underlay specific data model. DBr-UP maps the API-g to the API-s which is specific to ML underlay networks.

NOTE 1- API-g is defined by ML application provider and provided as a service by the ML pipeline. API-g depends on the data model used in the ML application. It also depends on the data used by the ML model which is used in the ML application. ML application provider specifies the data model to be used for the ML application, trains the ML model in the ML sandbox accordingly and then provisions the ML data model in the ML metadata store.

NOTE 2– Trained ML Models may be then published in ML model repositories.

8.1.2.3 API-s

API-s specifies the API which has to be used towards corresponding ML underlay networks. API-s is used by DBr-UP, in conjunction with API-g, to map the generic API to specific APIs of ML underlay networks.

NOTE 1- API-s is defined by ML underlay provider and provided as a service by the SRC and SINK. API-s depends on the data model used in the ML underlay. It also depends on the application which uses such data in the ML underlay. e.g., ML underlay specifies the data model to be used for the management application, provisions the element manager or VNFM accordingly and then provisions the data model in the data base, and lastly provisions third party management applications like OSS/BSS, orchestrators.

API-s may be implemented as loadable modules based on the API definitions exposed by the ML underlay networks. Some the capabilities of interworking between the data broker and the NF in the underlay may use API-s.

NOTE 2- In many cases, the mapping of this data model to actual technology specific network data is vendor-dependent. This mapping has to be handled by the ML data broker using API-s.

NOTE 3 – Similar to REQ-ML-INT-003 of [ITU-T Y.3172], in case SRC and SINK are implemented in ML underlay network functions, then DBr-UP and API-s instantiation may depend on deployment.

8.1.2.4 ML data broker control plane (DBr-CP)

This component is the control plane part of ML data broker, which controls the mapping of data between the ML underlay network to the SRC of ML pipeline and also the ML data output from SINK of ML pipeline to ML underlay network. This control is based on the data broker session which is established by the MLFO. Data broker control plane is responsible for interfacing between the MLFO and the DBr-UP. It configures the DBr-UP based on the decisions done by MLFO on the data model and corresponding mapping to ML underlay networks. This component uses interface 5.3 towards MLFO.

8.1.2.5 ML data broker user plane (DBr-UP)

This component is the user plane part of ML data broker, which is responsible for interfacing with the DBr-CP (data broker control plane) to facilitate the use of correct API-s towards the ML underlay network functions and to map the incoming data to the ML data model used in the ML overlay (if needed).

NOTE - Specific location of the DBr-UP in the technology specific underlay may depend on the constraints specified in the ML intent by the ML application author. For example, data input for the ML application may be from AN or CN or MEC, correspondingly the location of DBr-UP can be determined by MLFO so that the data requirements and latency budgets of the ML underlay can be met.

8.1.2.6 ML Data Base (DB)

This component provides the storage support for data used for ML. This storage may be used for sharing of data between various components of data handling framework. The schema of data storage may depend on the ML application and the components which are exchanging the data. It interfaces with MLFO using reference point 5.1. MLFO may control the interfaces, data models used in the ML DB, and the optimization of storage capacity using reference point 5.1.

NOTE - ML DB implementation may use a distributed or centralized approach.

8.1.3 Data handling framework supporting components

8.1.3.1 ML model repository

This supporting component is a repository of machine learning models.

NOTE - ML model repository may contain metadata which points to the data model used for training a given ML model.

8.2 High-level architecture

The data handling framework uses the ML intent to understand the nature of the ML application and the evolving network capability. While this helps in setting up the initial data handling reference points (e.g. SRC and SINK), the dynamic changes in the ML underlay is indicated by ML underlay orchestrator to the MLFO via the reference point 7 in [ITU-T Y.3172]. MLFO in turn uses reference point 5 to adapt the data handling framework components to handle the changes in the ML underlay. Examples of such changes include provisioning of new services in the ML underlay network (triggered by SO) or changes in the topology (triggered by VIM). The interface to orchestrator via reference point 7 allows the data handling framework to adapt to such changes in the ML underlay. An example scenario of addition of a new SRC is described in figure 8.3.

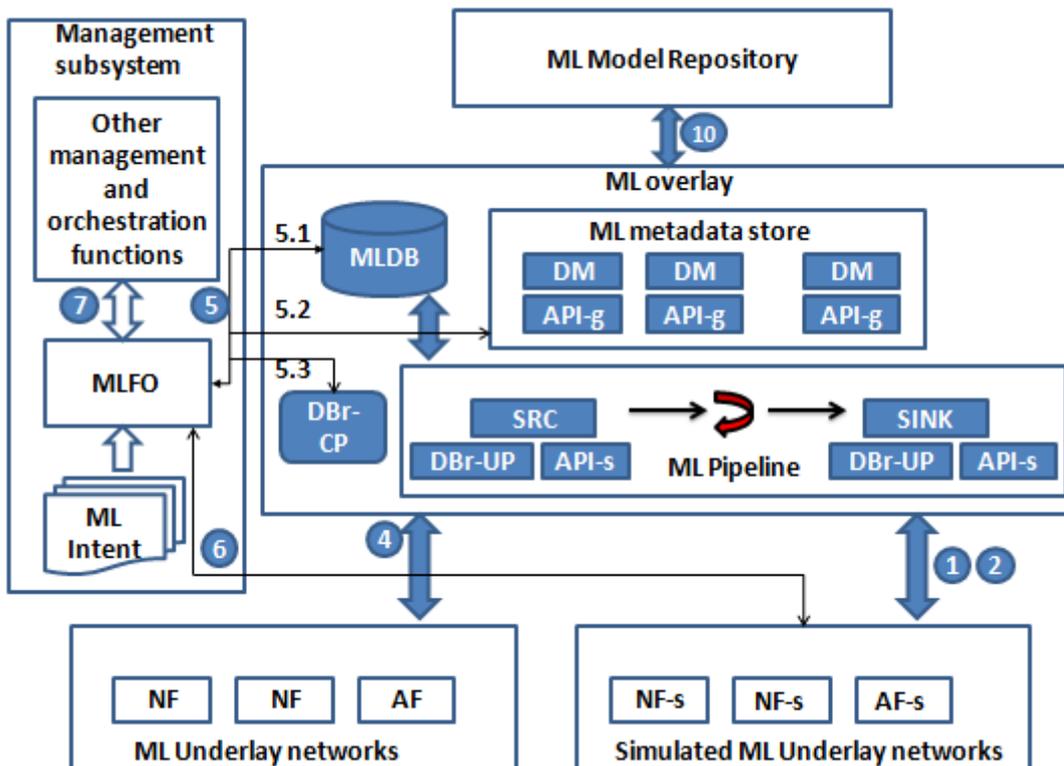


Figure 8.2 - High-level architecture of the data handling framework

The high-level architecture depicted in Figure 8-2 uses the following reference points described as follows:

- Reference points 1 (for SINK), and 2 (for SRC) defined in [ITU T Y.3172] are used as data handling interface for Simulated ML underlay networks.
- Reference point 4 defined in [ITU T Y.3172] is used as data handling interface for ML underlay networks.

NOTE 1- This reference point includes API-s which provides the interface for the data model specified by the ML underlay network.

- Reference point 5 is defined in [ITU T Y.3172] serves as interface between MLFO and ML pipeline subsystem. For the purposes of this Recommendation, this reference point is further decomposed as follows:
 - Reference point 5.1 is used between the MLFO and the ML DB. This reference point implements the requirements mentioned under ML-data-collection-storage mentioned in clause 7.1. In addition, as mentioned in clause 8.1.6, this reference point configures the ML DB with the data models and the optimizations in the data storage.
 - Reference point 5.2 is used between the MLFO and the ML metadata store. This reference point is used for query and management of data models in the ML metadata store.
 - Reference point 5.3 is used between the MLFO and the DBr-CP. This reference point is used for control of the mapping of data between the ML underlay and the ML overlay. This control is done by the MLFO by the creation of data broker sessions. The selected data model and the API-s to be used towards the ML underlay are signaled using reference point 5.3 by the MLFO to the DBr-CP.
- Reference point 6 defined in [ITU T Y.3172] is used as interface between MLFO and simulated ML underlay networks.
- Reference point 7 defined in [ITU T Y.3172] is used as interface between MLFO and management and orchestration functions of ML underlay networks, e.g. those defined in [ITU-T Y.3111].
- Reference point 10 is between ML model repository and ML metadata store. It is used to transfer ML data models.

NOTE 2 - Only the reference points between the components of the data handling framework are shown, e.g. those between the ML pipeline nodes are not shown in Figure 8-2.

NOTE 3 - As discussed in [ITU-T Y.3172], ML pipeline can be used in ML sandbox or in the live network. In both cases, the subsystems for data handling are the same. Hence Figure 8-2 depicts these deployments of ML pipeline, irrespective of they are ML sandbox or live network, in a common manner and are not shown separately in the figure.

8.3 Sequence diagrams for ML data handling

This clause shows the interaction of the framework components for various scenarios in the form of sequence diagrams for ML data handling in future networks including IMT-2020.

8.3.1 Instantiation of data handling framework

Figures 8-3.1 and 8-3.2 describe the instantiation of the data handling framework.

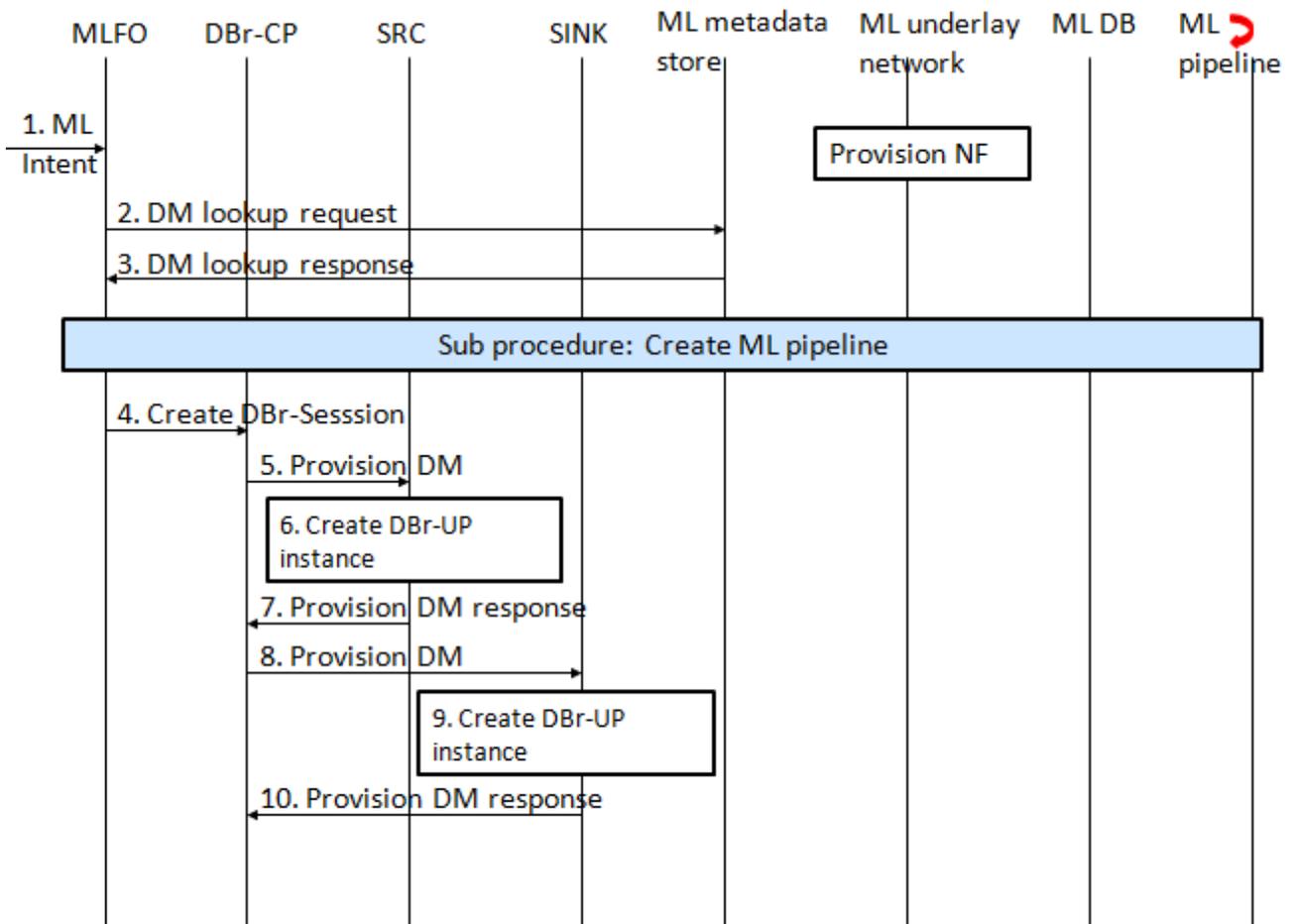


Figure 8-3.1 - Instantiation of data handling framework (part 1)

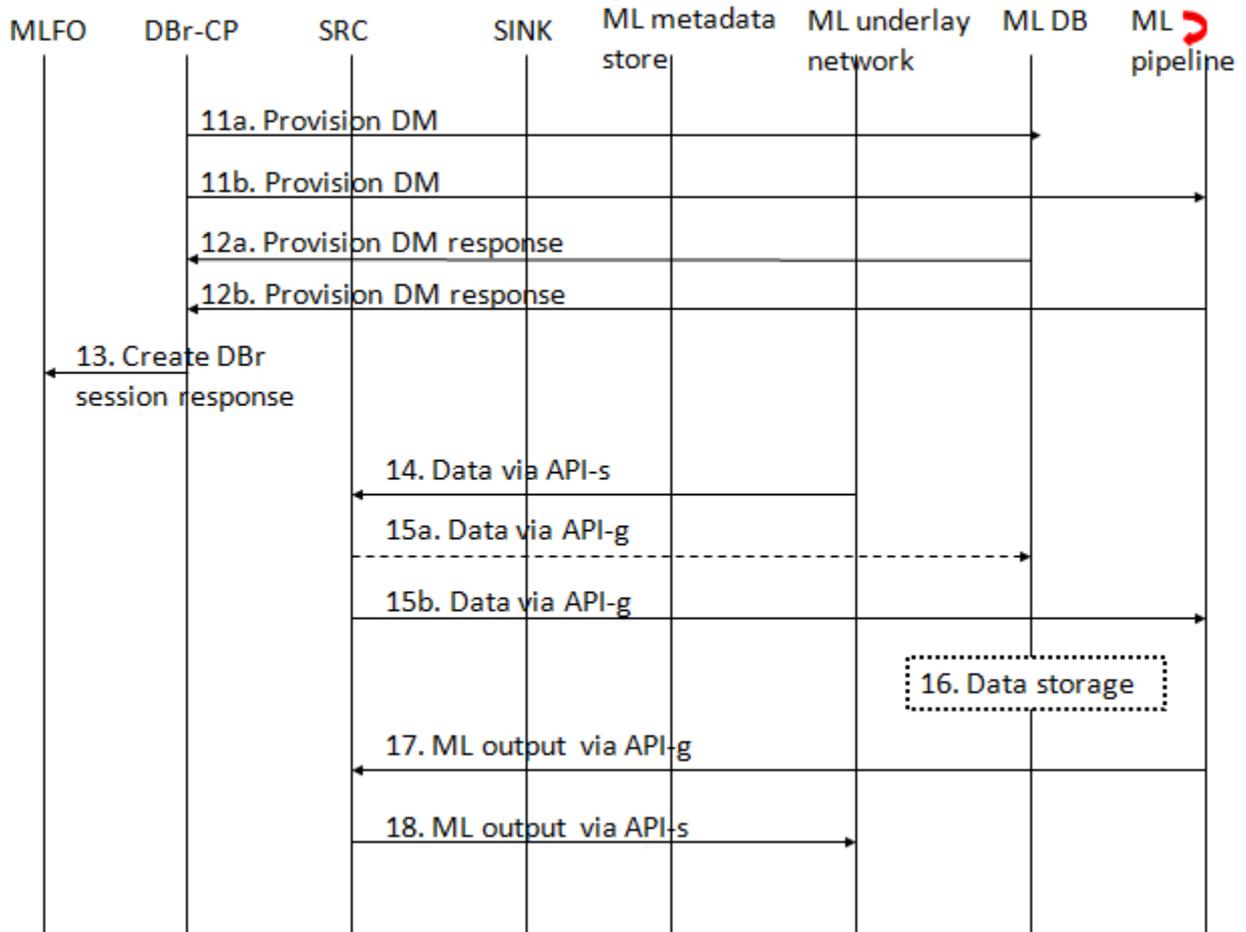


Figure 8-3.2 - Instantiation of data handling framework (part 2)

NOTE 1- It is assumed that the ML underlay provisions the NF according to the operator’s procedures, e.g., commissioning of base station.

NOTE 2- In case the ML underlay used is simulated, the ML pipeline is assumed to be in sandbox and used for training and testing. In case the ML underlay used is actual network (not simulated), then the ML pipeline is assumed to be in the ML overlay and used for inference in the network. The sequence diagram shown in Figures 8-3.1 and 8-3.2 includes the following steps:

1. ML Intent is provided as input to the MLFO. This includes the data elements to be used in the ML application, the data model which is proposed (if any) and any other inputs to MLFO.
2. MLFO looks up in the ML metadata store, the data model which satisfies the requirements of the ML application.
3. ML metadata store responds with a data model which fits the requirement.

NOTE 3- In this flow, it is assumed that a match is found in a standard DM and the message “3. DM lookup resp” in Figure 8-3.1 contains a reference to that standard DM. It is possible that there is no match found, that case is handled in Figure 8-5.

NOTE 4- At this point, a ML pipeline is created as per the specification of the ML application. This includes the creation of all the nodes specified in [ITU T Y.3172].

4. MLFO interfaces with DBr-CP to create a session for the ML application. It provides the DM to be used, the address of the SRC and SINK. In case there are multiple SRCs and/or SINKs timing sync requirements between them is specified by the MLFO in this step. Any latency constraint for the ML application in terms of data collection or configuration is used at this stage to configure the DBr-CP accordingly.
5. DBr-CP interfaces with the SRC to provision the DM. This also loads the corresponding API-g.
6. DBr-UP instance is created in the SRC. This also loads the corresponding API-s to be used towards the ML underlay network. This step configures the characteristics of the data collection, e.g., periodicity and storage characteristics.
7. SRC sends a response to DBr-CP upon creation of the DBr-UP instance. This message may include the characteristics of DBr-UP, e.g., the overhead introduced by it in terms of latency.
8. DBr-CP interfaces with the SINK to provision the DM. This also loads the corresponding API-g.
9. DBr-UP instance is created in the SINK. This also loads the corresponding API-s to be used towards the ML underlay.
10. SINK sends a response to DBr-CP upon creation of the DBr-UP instance. This message may include the characteristics of DBr-UP, e.g., the overhead introduced by it in terms of latency.
11. In step 11a: the data model is provisioned in the ML DB so that any storage of data can be done. This includes the metadata regarding policies and lifetime.

In step 11b: the data model is provisioned in the ML Pipeline so that any processing of data can be done in the ML pipeline.

12. In steps 12a and 12b, ML DB and ML pipeline sends Provision DM responses back to DBr-

CP.

13. At this point the DBr-CP may have an estimate of the overhead, e.g., the overhead introduced by it in terms of latency, introduced by the data handling. This may be given to the MLFO via an optional message (Create DBr Session Response).
14. Data specific to the ML underlay starts arriving at the SRC via API-s.
15. DBr-UP instance in the SRC maps the data to the API-g. The data then follows the chain created in the sub-procedure for creating ML pipeline. This may involve:
 - Step 15a: optionally sending the data to ML DB for storing
 - Step 15b: sending the data to the next node in the ML pipeline from the SRC.
16. ML DB stores any data according to policy. This is an optional step.
17. ML pipeline uses API-g to provide ML output to the SINK.
18. DBr-UP instance in the SINK maps the ML output to the API-s.

8.3.2 Addition of new SRC in data handling framework

Figures 8-4.1 and 8-4.2 describes the addition of new SRC in the data handling framework.

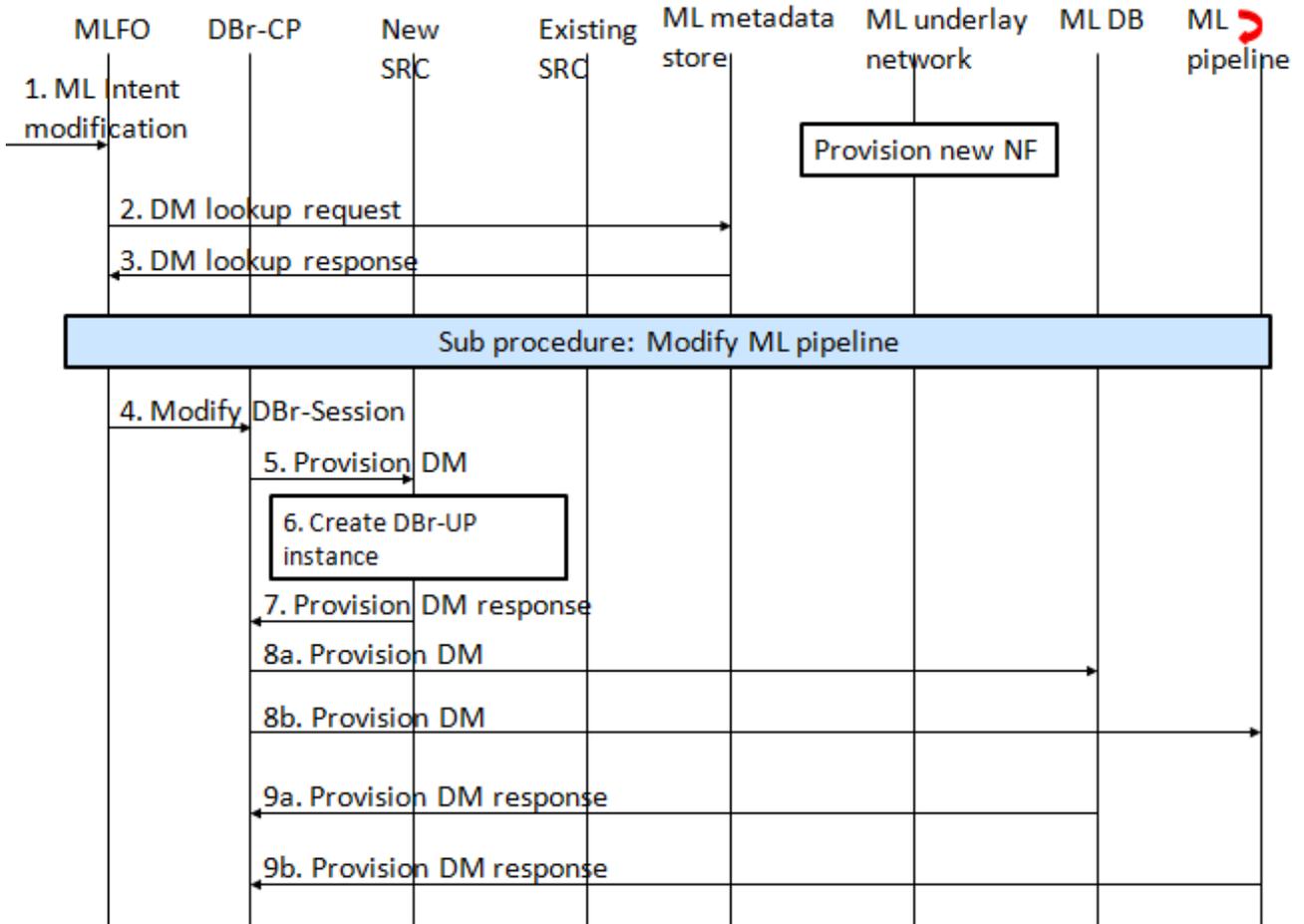


Figure 8-4.1 - Addition of new SRC in data handling framework (part 1)

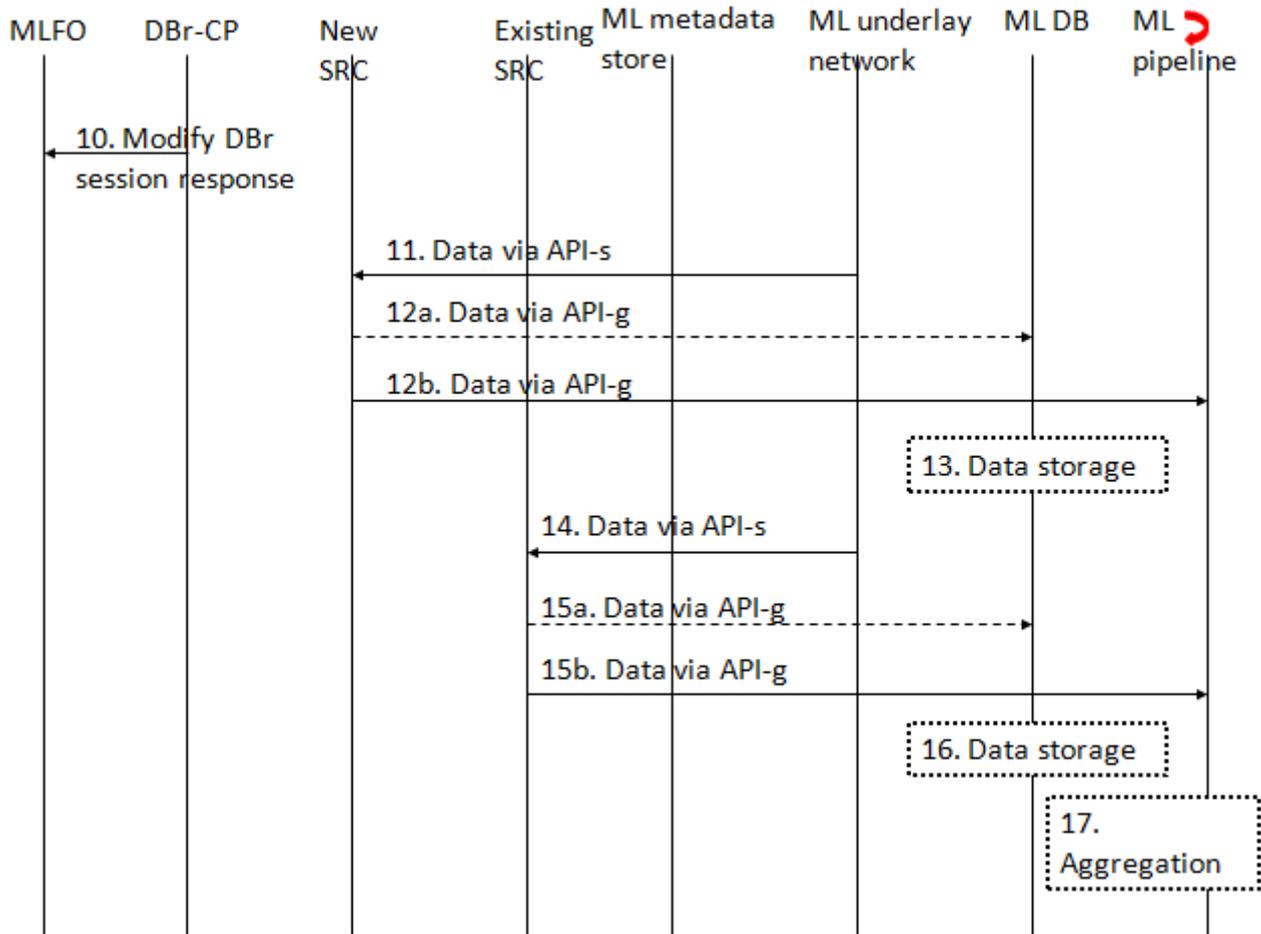


Figure 8-4.2 - Addition of new SRC in data handling framework (part 2)

NOTE 1- prerequisites: it is assumed that the data handling framework is already instantiated as shown in Figures 8-3.1 and 8.3.2. It is assumed that the new NF which acts as a source of data is provisioned in the ML underlay as per operator procedures.

The sequence diagram shown in Figures 8-4.1 and 8-4.2 includes the following steps:

1. ML Intent is modified as input to the MLFO. This includes, for the purposes of discussion on data handling, the data elements to be used in the new SRC, the data model which is proposed (if any).
2. MLFO looks up in the ML metadata store, the data model which satisfies the requirements of the new SRC.
3. ML metadata store responds with a data model which fits the requirement.

NOTE 2- In this flow, it is assumed that a match is found in a standard DM and the message “3. DM lookup resp” in Figure 8-4.1 contains a reference to that standard DM. It is possible that there is no match found, that case is handled in Figure 8-5.

4. MLFO interfaces with DBr-CP to modify the session for the ML application. It provides the DM to be used, the address of the new SRC.

NOTE 3- ML pipeline is assumed to be modified using another sub procedure, which is not the focus here. This may involve changes in chaining of ML overlay nodes to include the new SRC.

5. DBr-CP interfaces with the new SRC to provision the ML DM. This also loads the corresponding API-g.
6. DBr-UP instance is created in the new SRC. This also loads the corresponding API-s to be used towards the ML underlay (simulator in this case).

NOTE 4- it is assumed that no other changes except addition of the new NF in the ML underlay (which act as new SRC in the ML overlay).

7. The new SRC sends a response to DBr-CP upon creation of the DBr-UP instance. This message may include the characteristics of DBr-UP e.g. the overhead introduced by it in terms of latency.
8. The data model for the new SRC is provisioned in the framework.

Step 8a: This may include provisioning DM in the ML DB so that any storage of data can be done. This includes the metadata regarding policies and lifetime.

Step 8b: This may also include provisioning DM in the ML pipeline.

9. In steps 9a and 9b, the ML DB and ML pipeline send a response back to DBr-CP.
10. At this point the DBr-CP may have an estimate of the new overhead, e.g., the overhead introduced by it in terms of latency, introduced by the new SRC in data handling. This may be given to the MLFO via an optional message (Mod DBr Session Response).
11. Data specific to the ML underlay starts arriving at the new SRC via API-s.
12. DBr-UP instance in the new SRC maps the data to the API-g. The data then follows the chain

created in the sub-procedure for creating ML pipeline. This may involve:

Step 12a: optionally sending the data to ML DB for storing

Step 12b: sending the data to the next node in the ML pipeline from the SRC.

13. ML DB stores any data according to policy. This is an optional step.

14. Data specific to the ML underlay continues arriving at the existing SRC via API-s.

15. DBr-UP instance in the existing SRC maps the data to the API-g. The data then follows the chain created in the sub-procedure for creating ML pipeline. This may involve:

Step 15a: optionally sending the data to ML DB for storing

Step 15b: sending the data to the next node in the ML pipeline from the SRC.

16. ML DB stores any data according to policy. This is an optional step.

17. ML pipeline aggregates data from various SRCs. This is an optional step.

NOTE 5- ML pipeline provides the output for the ML underlay as described in Figure 8-3.2 steps 17 and 18.

8.3.3 Data model does not exist in the ML metadata store

Figure 8-5 describes the scenario where the ML intent refers to a new data model which is used by the ML application, but the data model is not known to the ML metadata store.

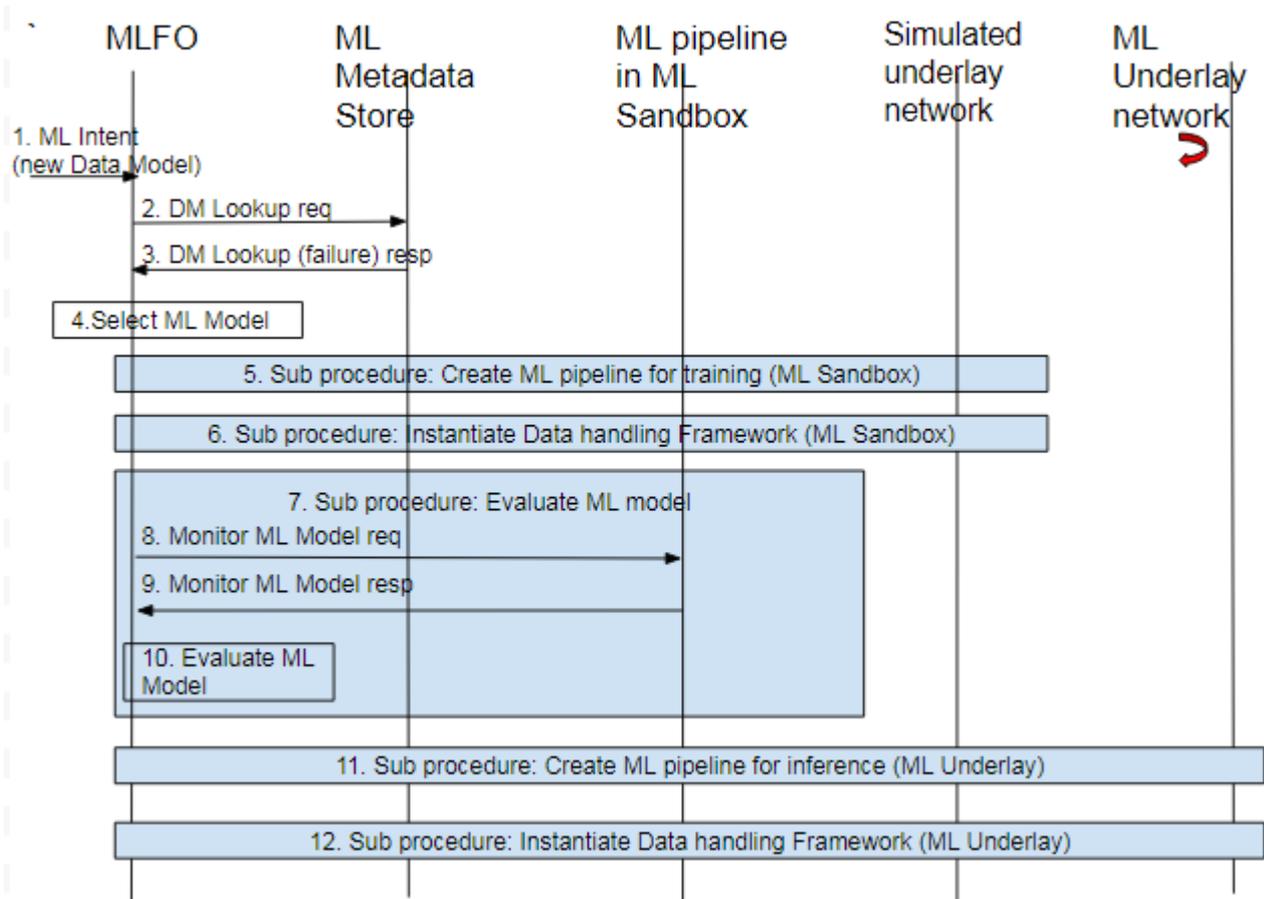


Figure 8-5 - Data model does not exist in the ML metadata store in the data handling framework

The sequence diagram shown in Figure 8-5 includes the following steps:

1. ML Intent is provided as input to the MLFO which has new data model being used by the SRC.
2. MLFO looks up in the ML metadata store.
3. DM lookup failure response is sent by the ML metadata store.

NOTE 1- The case of lookup failure is considered in this scenario to handle the situation where the data model is previously unknown to the ML metadata store.

4. Based on the requirements in the ML intent and the characteristics of the data, MLFO selects an untrained model.
5. MLFO creates an ML pipeline in the ML sandbox for training.

6. A data handling framework is deployed in the ML sandbox by the MLFO. This would be as shown in Figures 8-3.1 and 8-3.2.

NOTE 2- This step results in the simulator generating data and which is used for training the model.

7. A sub procedure for monitoring and evaluating the ML model is started. This may involve the following three steps.
8. MLFO monitors the ML model in the ML sandbox according to the requirements in the ML intent.
9. ML pipeline in the ML sandbox responds with the monitoring parameters.
10. The output from the monitoring is used to evaluate the ML model.
11. Based on the evaluation, an ML pipeline is deployed on the ML underlay. This includes the trained ML model being deployed in the ML pipeline corresponding to the ML underlay, to be used for inference.
12. A data handling framework is instantiated on the ML underlay as given in Figures 8-3.1 and 8-3.2.

9 Security considerations

The security considerations specified in [ITU-T Y.3172] are also applicable to this Recommendation.

Additional specific security considerations are provided in the requirements clause of this Recommendation.

Appendix I

Example realizations of Data Handling Framework

(This appendix does not form an integral part of this Recommendation.)

Figure I-1 gives an example of realization of the high-level data handling framework on an IMT-2020 network. [b-ITU-T Y.3104] [ITU-T Y.3111].

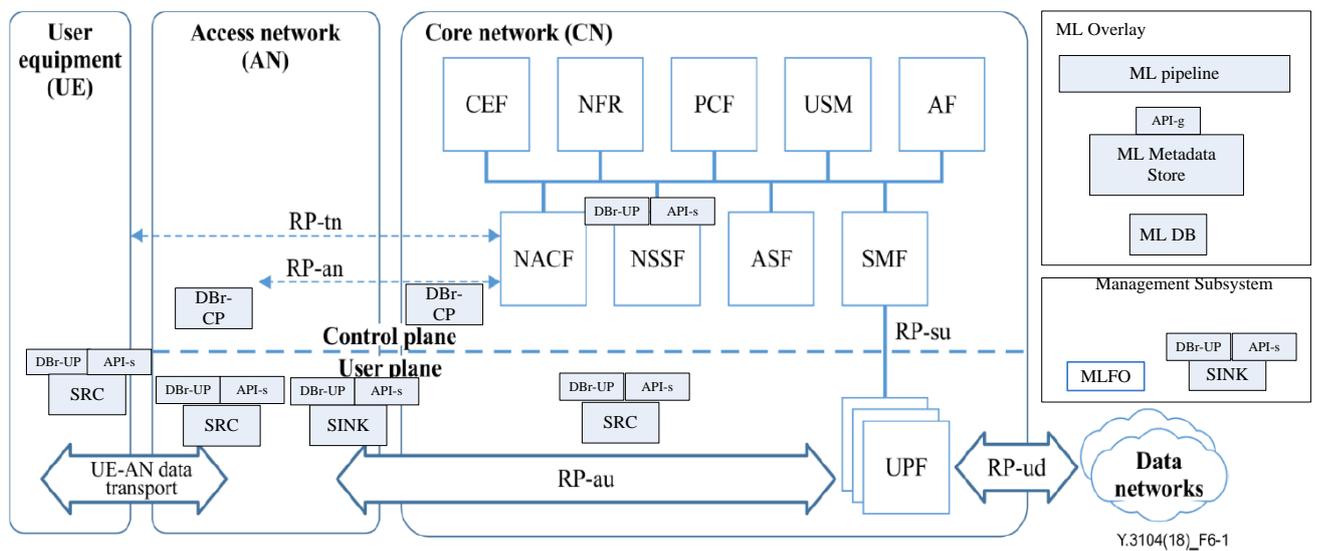


Figure I-1 Example of realization of the high level framework in an IMT-2020 network

The example of realization is represented in the following manner: The positions of the ML data base, ML data broker, ML metadata store and API-s and API-g are illustrated along with the underlay network.

Take the load balance and cell splitting/merging use case as an example, the SRC, data broker user plane and control plane can be deployed in the radio access network, the ML metadata store and the ML data base in the ML overlay and MLFO in the management subsystem. To facilitate the ML learning enabled load balance and cell splitting and merging, the load prediction and the mobility prediction are needed. The well-defined data models for the 3GPP network functions can be imported in the ML metadata store. SRC and SINK to be used for this ML application are specified by the ML intent input to the MLFO. The MLFO looks up the ML metadata store to check whether the required data model exists. If yes, MLFO creates a data broker session for this ML application. After

instantiating the data broker user plane and control plane, the SRC collects data from the radio access network via API-s and the corresponding data is provided to the ML pipeline deployed in the ML overlay, via API-g. The ML output for this ML applications, e.g., cell reselection parameters, and/or handover parameters, are delivered to the SINK in the ML pipeline via API-g and then to radio access network via API-s.

Appendix II

Mapping of requirements and capabilities of cloud computing based big data

(This appendix does not form an integral part of this Recommendation.)

The roles in cloud computing based big data system context are defined in [ITU-T Y.3600] such as:

- CSN:data provider;
- CSP:big data infrastructure provider;
- CSP:big data application provider;
- CSC:big data service user.

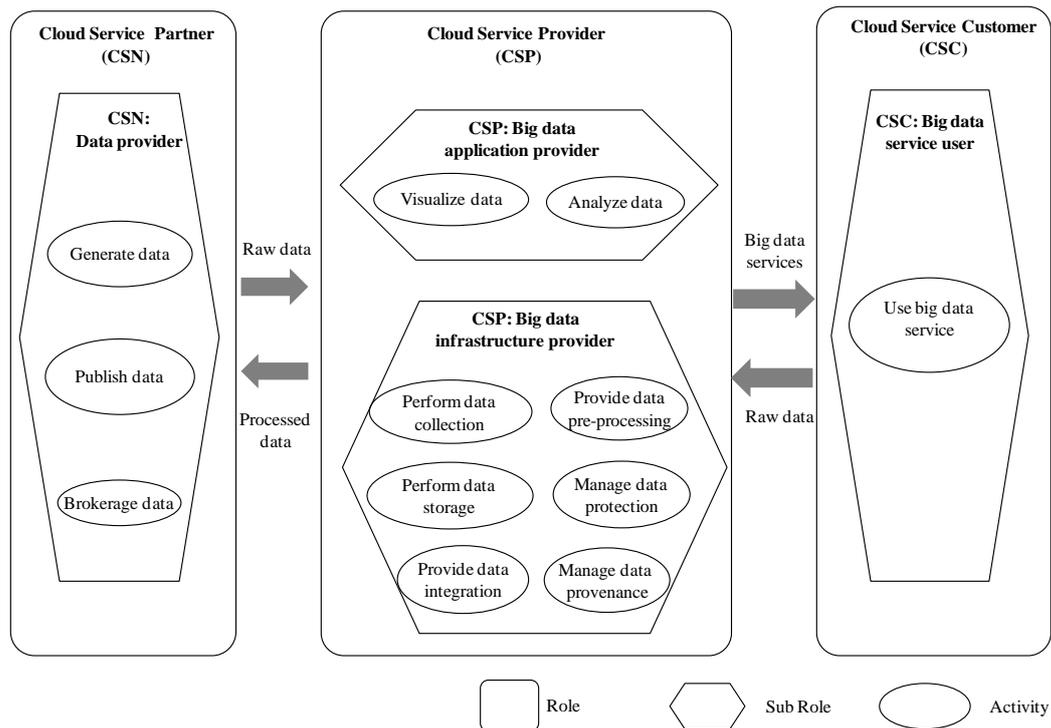


Figure II-1 – Cloud computing based big data system context [ITU-T Y.3600]

[b-ITU-T Y.3600] also defines the activities within the above. The data provider (CSN:DP) has activities: data generation, data publishing and data brokerage. Cloud service provider (CSP) has two sub-roles for big data analytics (CSP:BDAP) and infrastructure (CSP:BDIP). The big data service

user (CSC:BDSU) is the end-user or is a system that uses the results or services from a cloud service provider.

Table I.5 – Mobile network user behaviour big data analysis in [ITU-T Y.3600] describes a use case for cloud computing in support of big data, specifically in a mobile network. It further describes the possible scenarios of telecom operators to setup big data analytics in their network. Telecom operators can set up private big data infrastructure using cloud computing technologies or buy services from the cloud service provider: big data infrastructure provider (CSP:BDIP) and build big data application services or buy services from the cloud service provider: big data application provider (CSP:BDAP). Telecom operators could also act as big data service customer, using that for providing value-added services to end-users.

In case of the scenario described above, depending on the agreements with the CSP, telecom operators could choose to use the high-level architecture framework for ML in future networks including IMT-2020 as described in [ITU-T Y.3172] and the data handling framework described in this document, to instantiate the CSN:DP, CSP:BDIP, CSP:BDAP and the CSC:BDSU. This will facilitate the use of standardised ML pipeline based architecture and reference points described in [ITU-T Y.3172] for deployment scenarios described in Table I.5 of [ITU-T Y.3600]. Thus, the telecom operator is able to use uniform architecture in case of deployments of ML pipeline in future networks and IMT-2020, as well as cloud computing based big data analysis.

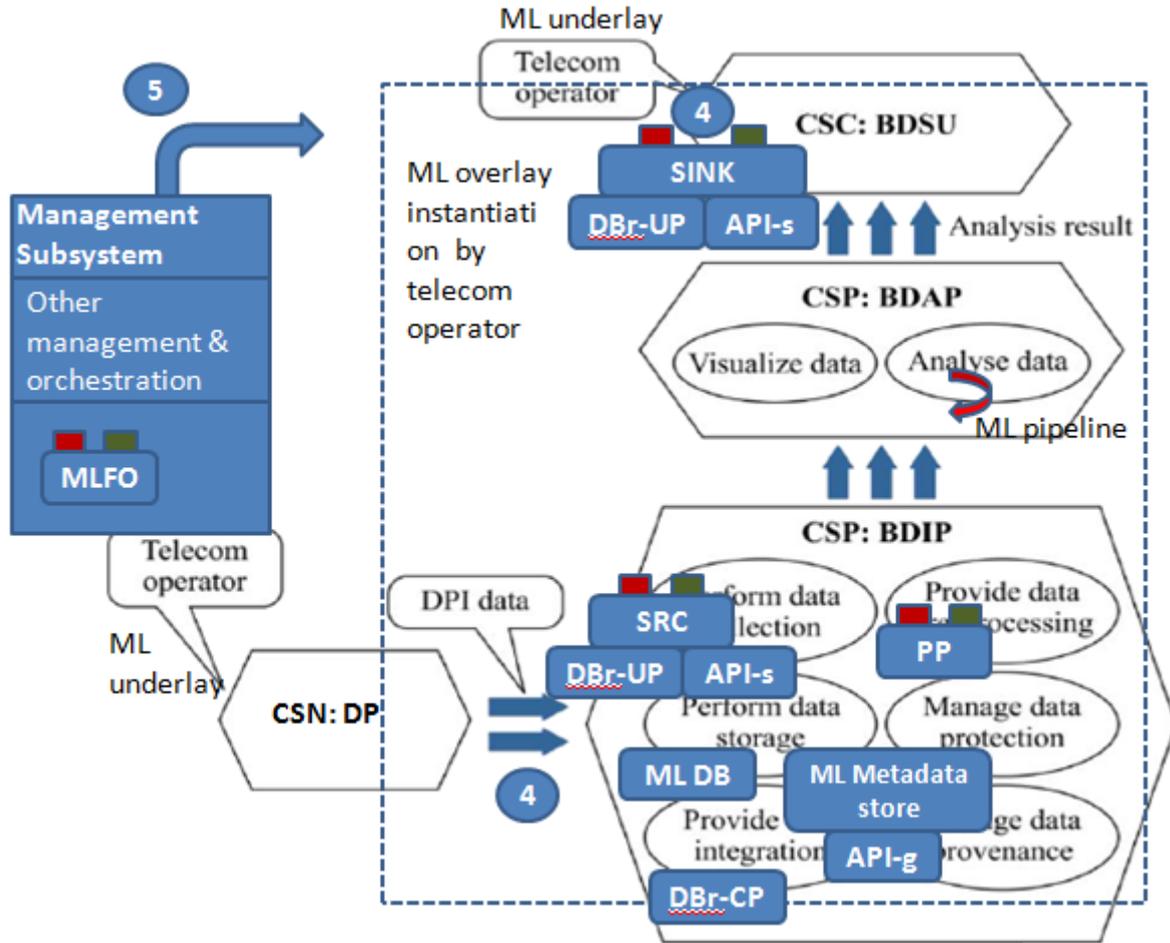


Figure II-2 - Instantiation of data handling framework for ML in cloud computing based big data

Figure II-1 describes the instantiation of data handling framework for ML in the cloud computing based big data under agreement between CSP and telecom operator. In such deployments, reference point 4 could use the data handling framework described in this document.

Table III-1 below provides a mapping of functionality between the data handling framework for ML and the services provided by cloud computing based big data.

NOTE - The mapping assumes that telecom operator may buy cloud services from CSP and build and/or customize the ML pipeline and data handling based on the terms of agreement with the CSP.

Table II-1 - Functionality mapping between data handling framework and cloud computing based big data

Functionality in this Recommendation	Mapping to terminology and services in cloud computing based big data
ML underlay	For the purposes of ML applications, telecom operators' networks would function as ML underlay networks. CSN:DP and CSC:BDSU in the cloud computing based big data are mapped to the SRC of data and the SINK for ML output respectively.
DBr-UP	The activities of the CSP:BDAP and CSP:BDIP can be mapped to these components, e.g. collecting data, storing data, integrating data, data analysis and data management.
API-s	
ML DB	
DBr-CP	
API-g	
ML metadata store	CSN:DP acts the corresponding functionality in the cloud computing based big data. The meta-information registry, a catalogue for searching usable data can be used for building the ML metadata store.
MLFO and reference point 5 [ITU-T Y.3172]	It is expected that the service orchestrator in the CSP:BDAP can be mapped to the reference point 5 as mentioned in [ITU-T Y.3172] in order to enable the MLFO to manage the ML overlay components.
Reference point 4 [ITU-T Y.3172]	The data interfaces between the CSN:DP, the CSP:BDAP and the CSC:BDSU, which are used for transfer of raw data, processed data and big data services, are mapped to the reference point 4.

Bibliography

- [b-3GPP 38201] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Physical layer; General description
- [b-3GPP 23501] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2 (Release 16)
- [b-ETSI 129 500] ETSI TS 129 500 V15.0.0 (2018-07) 5G; 5G System; Technical Realization of Service Based Architecture
- [b-ITU-T Supplement 55 to Y.3170-series] ITU-T Supplement 55 to Y.3170-series(2019), “Machine learning in future networks including IMT-2020: use cases”
- [b-ITU-T Y.3104] ITU-T Recommendation Y.3104 (2018), “Architecture of the IMT-2020 network”
- [b-ITU-T Y.3600] ITU-T Recommendation Y.3600 (2015), “Big Data-Cloud computing based requirements and capabilities”
-