International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Y.3174

(02/2020)

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Future networks

Framework for data handling to enable machine learning in future networks including IMT-2020

Recommendation ITU-T Y.3174

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Network control architectures and protocols	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	**Y.3000–Y.3499**
CLOUD COMPUTING	Y.3500–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	Y.4400–Y.4549
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.3174

Framework for data handling to enable machine learning in future networks including IMT-2020

Summary

Recommendation ITU-T Y.3174 describes a framework for data handling to enable machine learning in future networks including International Mobile Telecommunications (IMT)-2020. 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.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T Y.3174	2020-02-06	13	11.1002/1000/14134

Keywords

Architecture, data handling, IMT-2020, machine learning, requirements, training.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Recommendation ITU-T Y.3174

Framework for data handling to enable machine learning in future networks including IMT-2020

1 Scope

This Recommendation provides a framework for data handling to enable machine learning (ML) in future networks including International Mobile Telecommunications (IMT)-2020.

The scope of this Recommendation includes:

– background and motivations;

– high-level requirements of data handling and data models (DMs);

– framework for data handling to enable ML 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]	Recommendation ITU T Y.3111 (2017), *IMT-2020 network management and orchestration framework*.
[ITU-T Y.3172]	Recommendation ITU-T 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 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 [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.

3.1.4 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].

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

3.2.2 application programming interface-specific: A specific set of application programming interfaces for a machine learning data model (DM) to be used by a machine learning underlay network.

NOTE – Application programming interface-specific is used for specific operations on data in a machine learning underlay network (e.g., on collected data or on configuration management data in the underlay network).

3.2.3 machine learning database: A component which stores data related to machine learning in the machine learning overlay.

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

NOTE 1 – An ML data model may specify the data exchanged between an ML overlay and an ML underlay network.

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

3.2.5 machine learning data broker: A component which maps machine learning data models (DMs) from a machine learning underlay network to source and sink nodes of a machine learning pipeline.

3.2.6 machine learning metadata store: A component which stores machine learning data models (DMs) with their corresponding application programming interfaces.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AF	Application Function
AF-s	Simulated Application Function
API	Application Programming Interface
BDAP	Big Data Application Provider
BDIP	Big Data Infrastructure Provider
BDSU	Big Data Service User
CP	Control Plane
CSC	Cloud Service Customer
CSN	Cloud Service Partner
CSP	Cloud Service Provider
CU	Central Unit
DB	Database
DM	Data Model
DP	Data Provider
DU	Distributed Unit
IMT	International Mobile Telecommunications
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
RAN	Radio Access Network
SBA	Service Based Architecture
SDO	Standards Development Organization
SINK	Sink node
SO	Service Orchestrator
SRC	Source
UE	User Equipment
UP	User Plane

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 (ML) 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 governments. IMT-2020 networks, 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 to the need of handling large amounts of data in the network. [b-ITU-T Y.Sup.55] describes use cases where ML provides innovative and efficient solutions to problems in future networks including IMT-2020. A high-level architectural framework to enable ML in these future networks is also described in [ITU-T Y.3172].

The following three challenges are addressed in this Recommendation:

1) 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 architectural framework defined in [ITU-T Y.3172];
2) future networks are expected to be 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. This increased flexibility and agility makes it challenging to enable the use of ML mechanisms in future networks including IMT-2020;
3) evolution of networking techniques has resulted in evolving sources of data together with a multiplicity of 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 – Clause 8 of this Recommendation provides for an explanation on how these challenges are addressed using this data handling framework.

Building on the data handling reference points defined in [ITU-T Y.3172], the data handling framework for enabling ML in future networks including IMT-2020 includes ML data collection, ML data processing and ML data output. See Figure 6-1.

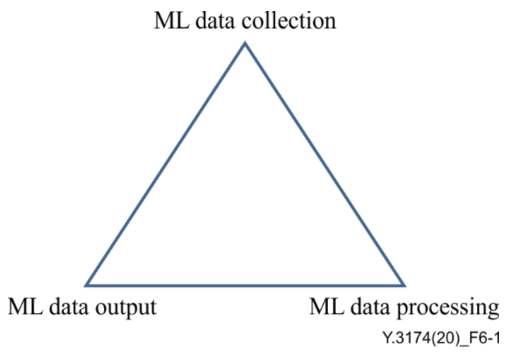


Figure 6-1 – Main concepts of data handling for ML

By analysing various use cases in [b-ITU-T Y.Sup.55], this Recommendation identifies a set of ML data collection requirements, ML data output requirements and ML data processing requirements. Then, taking into account these requirements, a framework for data handling is proposed for future networks including IMT-2020. The framework aims to solve the challenges previously listed and examples of 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 an implementation option.

7 High-level requirements for data handling

By analysing use cases described in [b-ITU-T Y.Sup.55], 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 support the storage and management of collected data.

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

NOTE 1 – Enabling 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 DMs used in the various ML pipeline nodes and the pre-processing 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 – A human operator may be able to query for which data have been deleted and the reason for deletion, e.g., the data have no more relevance 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 are 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 are valid to be used for an ML application with consideration of geographic locations.

REQ-ML-DH-009: ML data collection is recommended to optimize the deletion and archiving of stored data.

NOTE 6 – The optimization may be based on factors such as predictability of the data, relativity of the data, priority service data preservation and requirements of ML DMs.

REQ-ML-DH-010: ML data collection is required to enable the use of archived data, information or context at different levels of an 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 – Examples of network levels include access network network management systems (NMSs) and core networks.

NOTE 2 – 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 the use of application programming interfaces (APIs) 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 the collection of data from the operator hosted network functions (NFs) or from outside the operator hosted network functions.

NOTE 3 – Data sourced from 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., from a 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 electromagnetic (EM) properties, surface roughness and propagation mechanisms.

NOTE 5 – Additional examples of data sourced from outside the operator-hosted network include user equipment (UE) and base station (BS) location and location/mobility predictions, 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 and 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 neighbour cell, the number of radio resource control (RRC) connections requests in current cell and neighbour cell and 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 collected 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 the dynamic addition of SRCs associated with new data models.

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

REQ-ML-DH-021: ML data collection is required to be able to reuse existing interfaces to input data from network functions 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.Sup.55] derived from real and simulated environments.

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

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

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 such as antenna 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 key performance indicators (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 identification, 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 (UP) data along with relevant metadata.

NOTE 8 – Examples of metadata are 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 overhead of training, e.g., minimum sample size, which influences 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, and central and edge servers including their adaptive real time configurations.

NOTE 2 – ML data output may be used, for example, for 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 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, e.g., 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., an 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 – 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 data 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 not be affected by the methods used in managing ML data collection, ML data processing and ML data output.

7.2.4 ML-output-timing

REQ-ML-DH-047: ML data output is required to be bounded by the 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 data processing

7.3.1 ML-processing-models

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

7.3.2 ML-processing-data

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

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

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

7.3.3 ML-processing-levels

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

7.3.4 ML-processing-KPI

REQ-ML-DH-054: ML data 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 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

This clause describes a high-level framework for data handling which supports the requirements defined in clause 7. This includes the description of the framework components and sequence diagrams for data handling operations.

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

1) 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 examples of applying the data handling framework on technology-specific underlay networks.

8.1 Framework components and interfaces

This clause describes the high-level architectural components of the data handling framework for enabling the machine learning in future networks including IMT-2020. Integration of such components to an 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 architectural components of the data handling framework.

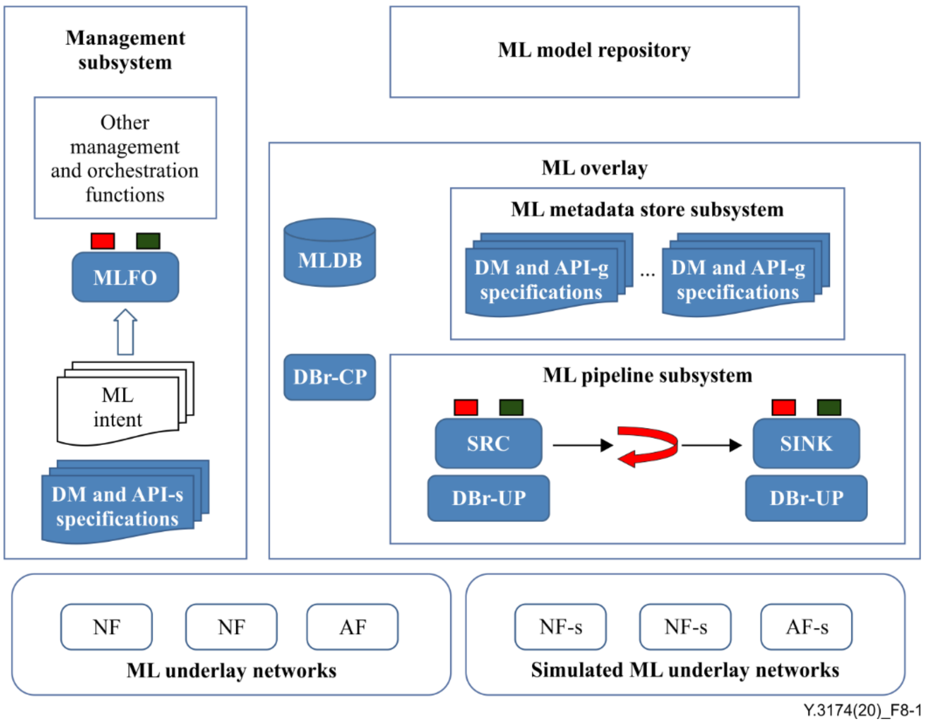

Management subsystem
Other management and orchestration functions
MLFO
ML intent
DM and API-s specifications
ML model repository
ML overlay
MLDB
ML metadata store subsystem
DM and API-g specifications
...
DM and API-g specifications
DBr-CP
ML pipeline subsystem
SRC
DBr-UP
SINK
DBr-UP
NF
NF
AF
ML underlay networks
NF-s
NF-s
AF-s
Simulated ML underlay networks
Y.3174(20)_F8-1

Figure 8-1 – High-level architectural components of the data handling framework

8.1.1 Reused components defined in ITU-T Y.3172

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

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

NOTE 1 – Simulated ML underlay networks are a part of ML sandbox.

NOTE 2 – Even though it is not shown in Figure 8-1, an ML sandbox can be used in combination with data handling framework to train and test machine learning models in the network. This is achieved by using ML pipeline deployment in the ML sandbox, simulated ML underlay networks and data handling reference points 1 and 2 as shown in Figure 4 in [ITU-T Y.3172].

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.

NOTE 3 – The newly defined data handling architectural component data broker user plane (DBr-UP) is instantiated in SRC and SINK nodes to map the data from and to the ML underlay networks. An instance of DBr-UP in SRC and SINK maps API-s (via reference point 4) to API-g used in DMs in the ML pipeline subsystem and vice versa.

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

8.1.2 Newly defined data handling architectural components

8.1.2.1 ML metadata store

The ML metadata store is an architectural component which stores the DMs usable in the network for ML applications. DM specifications present in the ML metadata store may have companion API specifications denoted as API-g as shown in Figure 8-2.

The purpose of such DMs includes application management, resource management and network management. DMs can be standardized or may be implemented with extensions and specific customizations. While specifying the ML applications, the use case designer can re-use any such standard DM, but the following challenges remain:

1) vendor implementations of such DMs in the ML underlay networks may differ;

2) a given ML application may need access to data from multiple levels used within 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 underlay networks. Thus, a unified view of DMs is needed for machine learning use cases.

The DMs specified for management purposes may not be sufficient for ML applications, nor be applicable to ML applications. In such cases, additional effort may be required to enable and map DMs from ML underlay networks to the ML DMs used in the ML overlay. Hence the DMs and corresponding API-g to be used for the ML applications are stored in the ML metadata store.

NOTE – The DMs stored in an ML metadata store are used as abstractions for data used across domains and across ML underlay networks for enabling ML applications. Specifications of DMs where available (e.g., from SDOs such as 3rd Generation Partnership Project (3GPP), Open Radio Access Network (O-RAN) Alliance, TM Forum (TMF)), may be imported and reused from ML model repositories.

8.1.2.2 API-g

The API-g is an architectural component which represents the API for a machine learning DMs to be used in a machine learning (ML) overlay. By defining it, ML data processing can be implemented without depending on the underlay specific DM. 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 DM 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 DM to be used for the ML application, trains the ML model in the ML sandbox accordingly and then provisions the ML DM in the ML metadata store.

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

8.1.2.3 API-s

The API-s is an architectural component which represents 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 the specific APIs of ML underlay networks.

NOTE 1 – API-s is defined by ML underlay network provider and provided as a service by the SRC and SINK nodes, i.e., instantiated in these ML pipeline nodes. API-s depends on the DM used in the ML underlay network. It also depends on the application which uses such data in the ML underlay network. e.g., the ML underlay network specifies the DM to be used for the management application, provisions the element manager or virtualised network functions manager (VNFM) accordingly and then provisions the DM in the ML database, and lastly provisions third party management applications like operations support system/business support system (OSS/BSS), orchestrators. Hence the DMs and corresponding API-s to be used towards the ML underlay network are stored in the management subsystem.

API-s may be implemented as loadable modules based on the API definitions exposed by the ML underlay networks. Interworking between the data broker (DBr-UP) and the NF in the ML underlay network may use API-s.

NOTE 2 – In many cases, the mapping of DM used in the ML overlay to the DM used in the ML underlay is implementation-dependent. This mapping has to be handled by the ML data broker (DBr-UP) using API-s.

NOTE 3 – Instantiation of DBr-UP and API-s may follow the same approach as for SRC and SINK nodes as mentioned in REQ-ML-INT-003 in [ITU-T Y.3172].

8.1.2.4 ML data broker control plane

The ML data broker control plane (DBr-CP) is an architectural component supporting functionalities of an ML data broker which controls the mapping of data between an ML underlay network to SRC nodes of an ML pipeline and also the ML data output from SINK nodes of the ML pipeline to the ML underlay network. This control is managed by by the MLFO. DBr-CP is responsible for configuring involved DBr-UPs based on the decisions taken by the MLFO regarding the DM and corresponding mapping to ML underlay networks. This architectural component uses interface 5.3 towards MLFO.

8.1.2.5 ML data broker user plane

The ML data broker user plane (DBr-UP)s is an architectural component which facilitates the use of appropriate API-s towards the ML underlay network functions and maps the incoming data from the ML underlay network functions to the ML DM used in the ML overlay.

NOTE – The location of a given DBr-UP in a technology specific underlay network may depend on the constraints specified in the ML intent by the ML application designer. For example, data inputs for a given ML application may be coming from AN, CN or MEC platforms, and correspondingly, the location of required DBr-UPs can be determined by the MLFO so that the data requirements including latency budgets can be met by the ML underlay networks.

8.1.2.6 ML database

The ML database (DB) is an architectural component providing storage support for data used for ML applications. This storage may be used for sharing data between various architectural components of the data handling framework. The schema of data storage may depend on the ML application and the components that are exchanging the data. The ML DB component interfaces with MLFO using reference point 5.1. MLFO may control the interfaces (API-g and API-s), DMs used by the ML DB for storage of data and the optimization of storage capacity using reference point 5.1.

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

8.1.3 Data handling framework supporting aspects

8.1.3.1 ML model repository

This is a repository of machine learning models.

NOTE – The ML model repository may contain metadata which points to the DMs used for training a given ML model.

8.2 High-level architecture

Figure 8-2 shows the high-level architecture of the data handling framework. This architecture builds upon the architectural components described in clause 8.1.

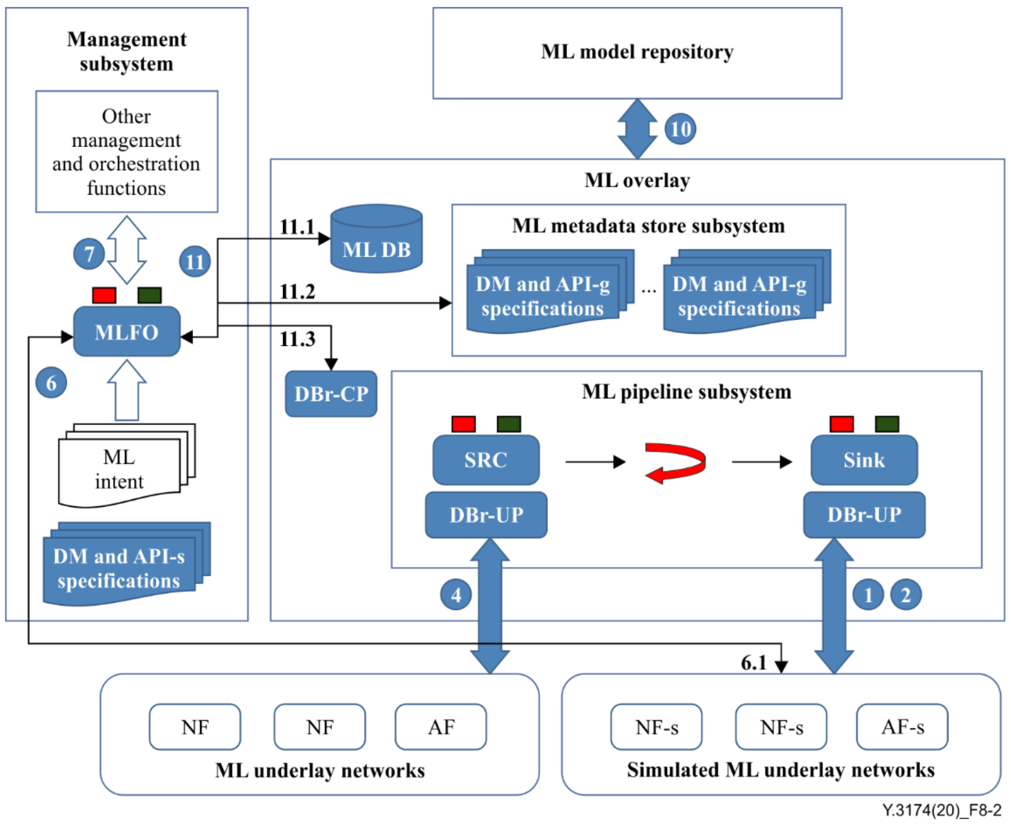


Figure 8-2 – High-level architecture of the data handling framework

The data handling framework uses the ML intent to understand the nature of the ML applications including the DMs to be used. While this helps in setting up the initial data handling reference points (e.g., with SRC and SINK nodes), the dynamic changes in the ML underlay networks are indicated by orchestrators of these ML underlay networks to the MLFO via the reference point 7 in [ITU-T Y.3172]. The MLFO in turn uses reference point 5 in [ITU-T Y.3172] to adapt the data handling framework components to handle the changes in the ML underlay networks. Examples of such changes include provisioning of new services in the ML underlay network (triggered by the service orchestrator (SO)) or changes in the topology (triggered by the virtualised infrastructure manager (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.

The high-level architecture depicted in Figure 8-2 uses the 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 reference points between simulated ML underlay networks and ML overlay;

NOTE 1 – This reference point uses API-s which provides the interface for the DM specified by the simulated ML underlay network.

– reference point 4 defined in [ITU-T Y.3172] is used as the data handling reference point between ML underlay networks and ML overlay;

NOTE 2 – This reference point uses API-s which provides the interface for the DM specified by the ML underlay network.

– reference point 11 serves as the interface between the MLFO and the data handling components in the ML overlay. This reference point is further decomposed as follows:
    - reference point 11.1 is used between the MLFO and the ML DB. This reference point supports the requirements under ML-data-collection-storage in clause 7.1. In addition,

as mentioned in clause 8.1.2.6, this reference point is used to provision the ML DB with the DMs and to manage the optimizations in the data storage;

  - reference point 11.2 is used between the MLFO and the ML metadata store. This reference point is used by the MLFO for querying and managing DMs in the ML metadata store;
  - reference point 11.3 is used between the MLFO and the DBr-CP. This reference point is used by the MLFO for controlling the mapping of data between the ML underlay networks and the ML overlay. This control is performed by the MLFO through the use of data broker sessions. The selected DM and the API-s to be used towards the ML underlay networks are signalled by the MLFO to the DBr-CP using reference point 5.3.

– reference point 6 defined in [ITU T Y.3172] as the interface between the MLFO and ML sandbox subsystem is decomposed here as point 6.1. Reference point 6.1 is used between MLFO and simulated ML underlay networks;

– reference point 7 defined in [ITU T Y.3172] is used as the interface between the MLFO and the management and orchestration functions of ML underlay networks, e.g., those defined in [ITU-T Y.3111];

– reference point 10 is between the ML model repository and ML metadata store. It is used to transfer ML DMs.

NOTE 3 – 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 4 – As discussed in [ITU-T Y.3172], ML pipeline can be used in an 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 illustrates the interaction between the framework components for various scenarios in the form of sequence diagrams.

8.3.1 Instantiation of data handling framework

Figures 8-3-a and 8-3-b describe the instantiation of a data handling framework.

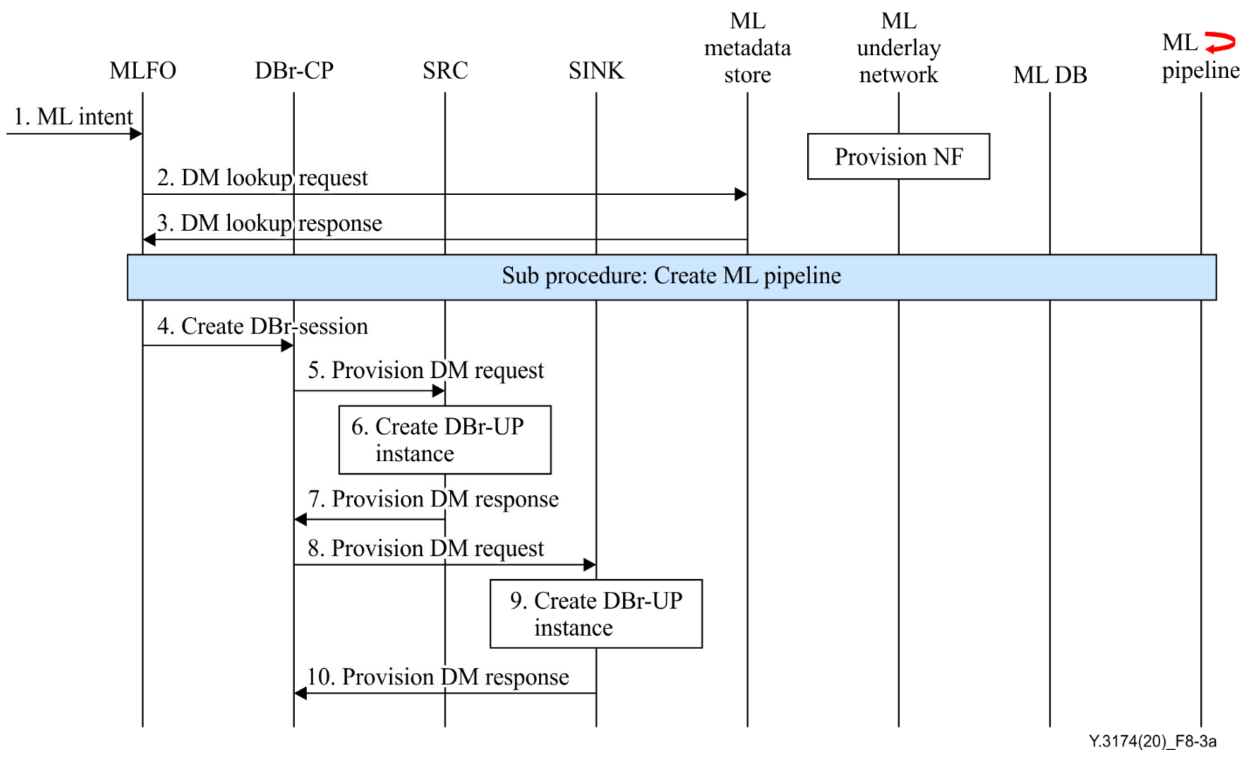


Figure 8-3-a – Instantiation of a data handling framework (part a)

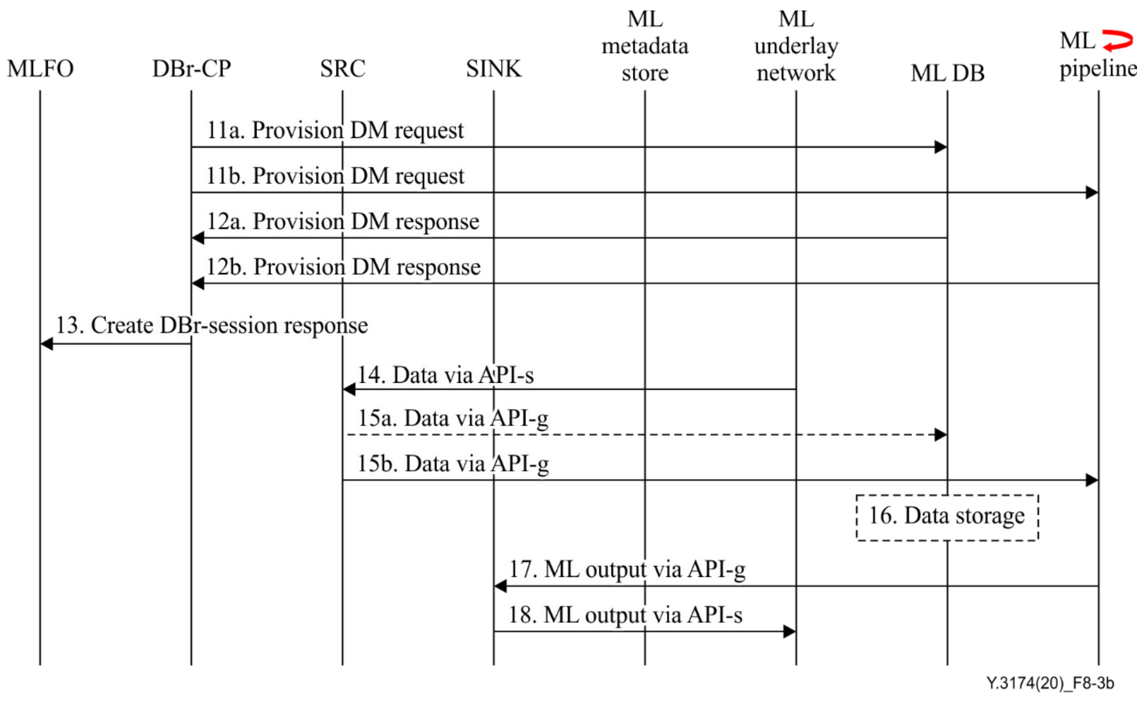


Figure 8-3-b – Instantiation of data handling framework (part b)

NOTE 1 – It is assumed that the ML underlay network has been provisioned with necessary NFs according to the network operator's procedures, e.g., commissioning of base station.

NOTE 2 – In case the ML underlay network is a simulated one, the ML pipeline is assumed to be in the ML sandbox and used for training and testing. In case the ML underlay network is a live network (not simulated), then the ML pipeline is assumed to be in the ML overlay and used for inference in the network. The sequence diagrams shown in Figures 8-3-a and 8-3-b include the following steps:

1) the ML intent is provided as input to the MLFO. This includes the data elements to be used in the ML application, the DM which is proposed (if any) and any other inputs to the MLFO;
2) using reference point 11.2, the MLFO looks up in the ML metadata store, the DM which satisfies the requirements of the ML application;
3) the ML metadata store responds through reference point 11.2 with a DM that matches to the ML application requirements.

NOTE 3 – In this flow, it is assumed that a match is found with a standard DM and that therefore the message "DM lookup response" in Figure 8-3-a contains a reference to that standard DM. The case where no match is found handled in Figure 8-5.

NOTE 4 – At this point, an ML pipeline is created under the control of the MLFO as per the specification of the ML application. This includes the instantiation of all the required ML pipeline nodes as specified in [ITU-T Y.3172].

4) the MLFO interacts with the DBr-CP using reference point 11.3 to create a session for the ML application. In the Create DBR-session request sent to the DBr-CP, the MLFO provides information about the DM to be used and the address of the SRC and SINK nodes. In case there are multiple SRCs and/or SINKs, timing synchronization requirements between them may be also specified by the MLFO in this step. Any latency constraints for the ML application in terms of data collection or configuration is used at this stage to configure the DBr-CP accordingly;
5) the DBr-CP interacts with the SRC node to provision the DM using the internal reference point 8. This interaction also allows loading the associated API-g to the SRC node;
6) after the receipt of the "Provision DM request", the SRC node creates a DBr-UP instance. This step also instantiates the corresponding API-s to be used towards the ML underlay network. This step is also used to configure the characteristics of the data collection, e.g., periodicity and storage characteristics;
7) following the creation of the DBr-UP instance, the SRC node sends a confirmation response message to the DBr-CP. The response message may include the characteristics of the created DBr-UP instance, e.g., the overhead introduced by the DBr-UP instance in terms of latency;
8) the DBr-CP interacts with the SINK node to provision the DM using the internal reference point 8. This step also instantiates the associated API-g in the SINK node;
9) after the receipt of the "Provision DM request", the SINK node creates a DBr-UP instance. This step also instantiates the corresponding API-s to be used towards the ML underlay network;
10) following the creation of the DBr-UP instance, the SINK node sends a confirmation response message to the DBr-CP. The response message may include the characteristics of the created DBr-UP instance, e.g., the overhead introduced by the DBr-UP instance in terms of latency;
11) step 11 includes two parts (not shown separately). First, in step 11a, the DM is provisioned by the MLFO in the ML DB so that any storage of data received from the SRC node using API-g can be done (cf. step 15 a). This step provisions also the metadata regarding policies and lifetime for the data stored in the ML DB. Second, in step 11b: the DM is provisioned in the ML pipeline so that any processing of data received from the SRC node(s) can be done in the ML pipeline (cf. step 15b);

12) step 12 includes two parts (shown separately). In step 12a, the ML DB sends a "Provision DM" confirmation response back to the DBr-CP. In step 12b, the ML pipeline sends a "Provision DM" confirmation response back to the DBr-CP;

13) at this point, the DBr-CP may have an estimate of the overhead, e.g., the overhead introduced by the data handling (cf. indications potentially received from DBr-UP instances in steps 7 and 10). This estimation may be optionally provided to the MLFO via the "Create DBr Session" response;

14) data specific to the ML underlay networks starts arriving at the SRC node via API-s;

15) the DBr-UP instance in the SRC node maps the received data received to API-g (communicated by MLFO in step 5). This step involves:

   – step 15a: the SRC node optionally sends the received data via API-g to the ML DB for storing;

   – step 15b: the SRC node sends the received data via API-g to the next node in the ML pipeline.

16) the ML DB stores any received data according to policies communicated by the MLFO in step 11. This is an optional step;

17) the ML pipeline uses API-g provided by the MLFO (cf. step 8) to provide the ML output data to the SINK node;

18) the DBr-UP instance in the SINK node maps the received ML output data to the ML underlay network using API-s.

8.3.2 Addition of new SRC in data handling framework

Figures 8-4-a and 8-4-b describe the addition of a new SRC node in the data handling framework.

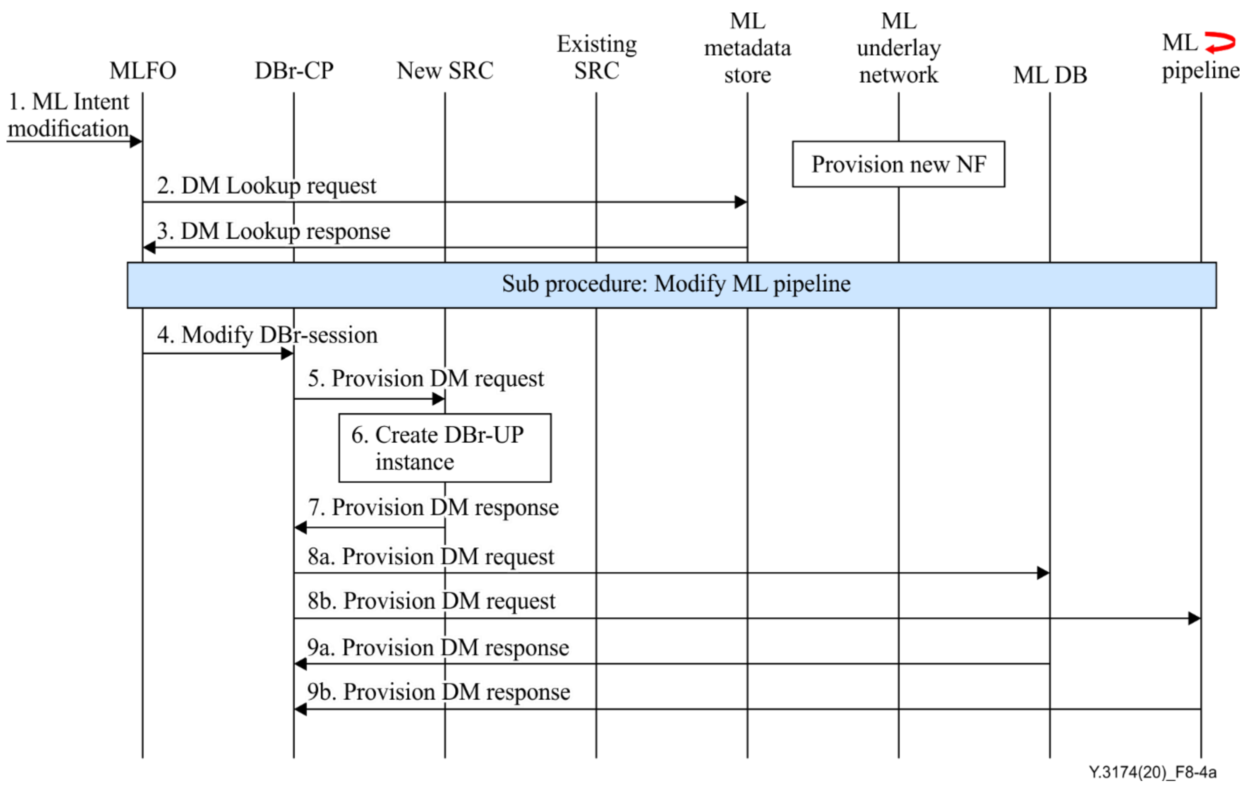


Figure 8-4-a – Addition of new SRC in data handling framework (part a)

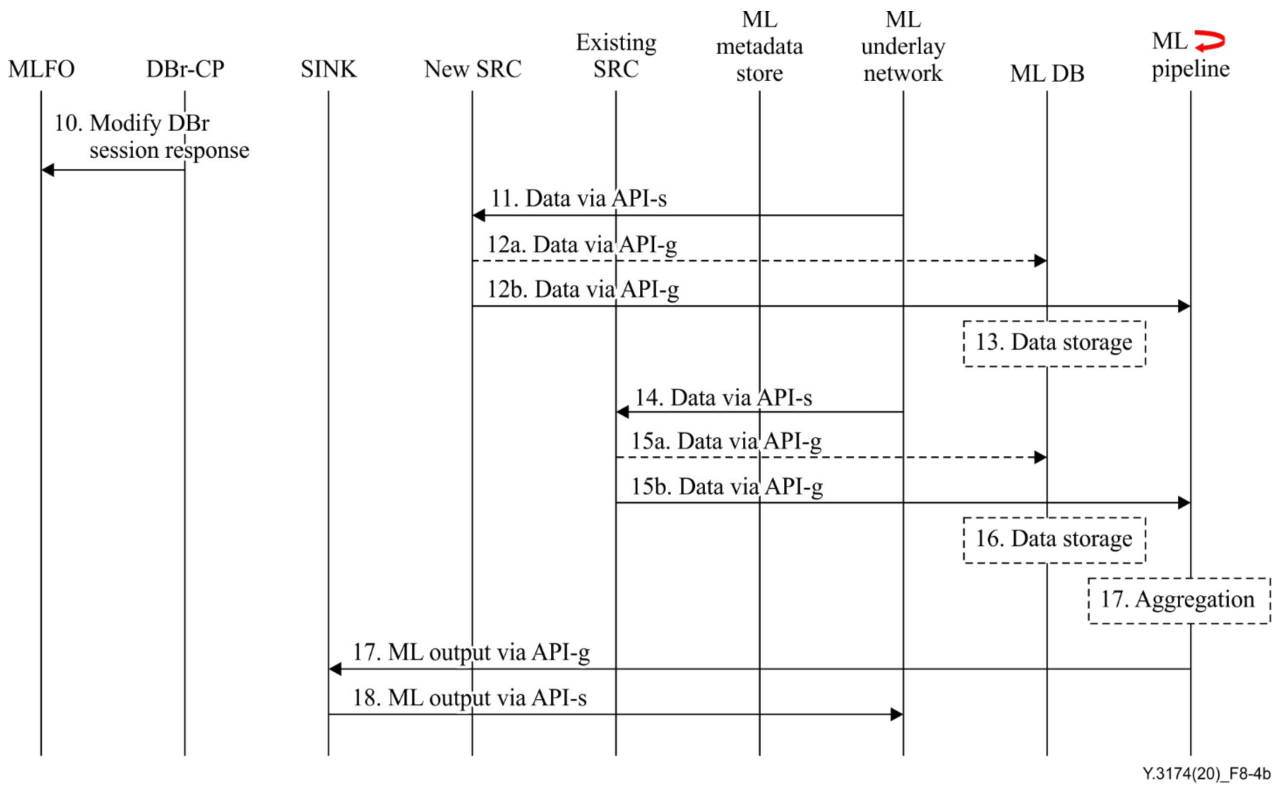


Figure 8-4-b – Addition of new SRC in data handling framework (part b)

NOTE 1 – As prerequisites, it is assumed that the data handling framework is already instantiated as shown in Figures 8-3-a and 8-3-b. It is also assumed that the NF which acts as a new source of data has been provisioned in the ML underlay network as per network operator procedures.

The sequence diagrams shown in Figures 8-4-a and 8-4-b include the following steps:

1) the modified ML intent is provided as input to the MLFO. This ML intent includes the data elements to be used in the new SRC node and the suggested DM to be used;

2) the MLFO queries the ML metadata store for a DM which satisfies the requirements of the new SRC node;

3) the ML metadata store responds with a DM which matches the requirements communicated by the MLFO;

NOTE 2 – In this flow, it is assumed that a match is found with a standard DM and that therefore the message "DM lookup response" in Figure 8-4-a contains a reference to that standard DM. The case where no match is found handled in Figure 8-5.

4) the MLFO interacts with the DBr-CP to modify the established session for the ML application. It provides the new DM to be used as well as the address of the new SRC node to be added;

NOTE 3 – The ML pipeline is assumed to be modified using another procedure, which is not detailed here. This procedure may involve changes in the chaining of ML pipeline nodes to include the new SRC node.

5) the DBr-CP interacts with the new SRC node to provision the ML DM. This step also instantiates the corresponding API-g;

6) a DBr-UP instance is created in the new SRC node. This step also instantiates the corresponding API-s to be used towards the ML underlay network;

NOTE 4 – It is assumed that there are no other changes in the ML underlay network except the addition of a new NF which acts as new SRC node in the ML overlay.

7) the new SRC node sends a response to the DBr-CP upon creation of the DBr-UP instance. This message may include the characteristics of the created DBr-UP instance, e.g., the overhead introduced by this instance in terms of latency;

8) the DM for the new SRC node 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 (cf. step 15a). This includes the metadata regarding policies and lifetime;
   – step 8b: this may also include provisioning DM in the ML pipeline (cf. step 15b).

9) in step 9a, the ML DB sends a confirmation response back to the DBr-CP. In step 9b, the ML pipeline sends a confirmation response back to the DBr-CP;

10) at this point the DBr-CP may have an estimate of the new overhead, e.g., the overhead introduced by the newly created DBr-UP instance in terms of latency, introduced by the new SRC node in data handling. This estimate may optionally be given to the MLFO via the "Modify DBr Session" response;

11) data specific to the ML underlay network starts arriving at the new SRC node via API-s;

12) the DBr-UP instance in the new SRC node maps the received data to the API-g. This involves:
   – 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) the ML DB stores any data according to the data retention policies. This is an optional step;

14) data specific to the ML underlay network continues arriving at the other already active/running SRC nodes via API-s;

15) the DBr-UP instances in the other already active/running SRC nodes map received data to the API-g. This involves:
   – 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.

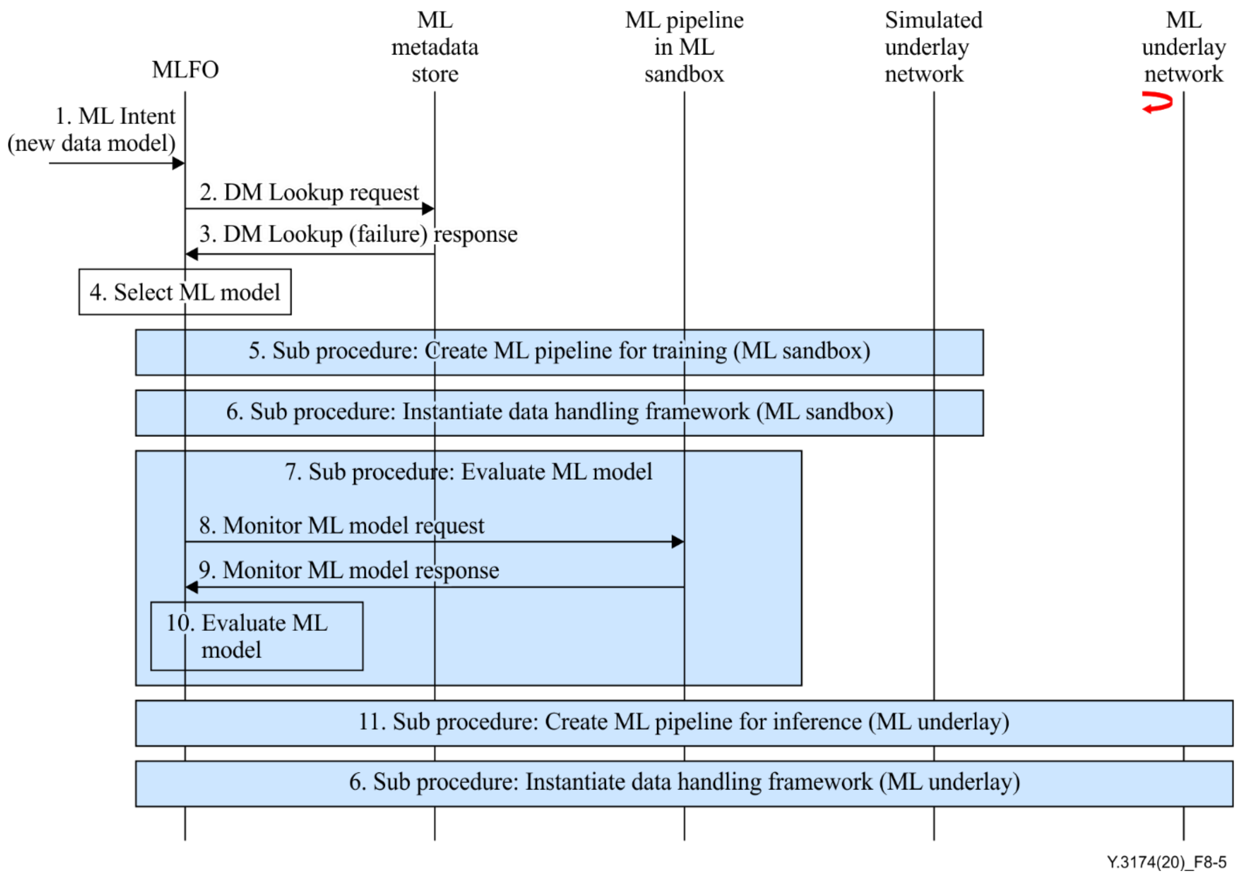
16) the ML DB stores any data according to the data retention policies. This is an optional step;

17) the ML pipeline uses API-g provided by the MLFO (cf. step 8) to provide the ML output data to the SINK node;

18) the DBr-UP instance in the SINK node maps the received ML output data to the ML underlay network using API-s.

8.3.3 DM not present in the ML metadata store

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



MLFO
ML metadata store
ML pipeline in ML sandbox
Simulated underlay network
ML underlay network
1. ML Intent (new data model)
2. DM Lookup request
3. DM Lookup (failure) response
4. Select ML model
5. Sub procedure: Create ML pipeline for training (ML sandbox)
6. Sub procedure: Instantiate data handling framework (ML sandbox)
7. Sub procedure: Evaluate ML model
8. Monitor ML model request
9. Monitor ML model response
10. Evaluate ML model
11. Sub procedure: Create ML pipeline for inference (ML underlay)
6. Sub procedure: Instantiate data handling framework (ML underlay)
Y.3174(20)_F8-5

Figure 8-5 – DM not known in the ML metadata store

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

1) the SRC in the ML intent uses a new DM (not present in the ML metadata store);

2) the MLFO queries the ML metadata store for the DM;

3) a 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 DM is not present in the ML metadata store.

4) based on the requirements provided in the ML intent and the characteristics of the input data from the SRC for the ML application, the MLFO selects an untrained ML model;

5) the 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 (i.e., according to the sequence diagram described in clause 8.3.1).

NOTE 2 – This step results in the simulated ML underlay network generating data which is used for training the model.

7) a procedure for monitoring and evaluating the ML model is started. This may involve the following three steps:

   a) the MLFO monitors the ML model in the ML sandbox according to the requirements in the ML intent;

   b) the ML pipeline in the ML sandbox provides a reporting of the monitored parameters;

   c) the output from the monitoring is used to evaluate the ML model.

8) based on the evaluation, an ML pipeline is deployed on the ML underlay network. This deployment includes the trained ML model in the ML pipeline;

9) a data handling framework is instantiated on the ML underlay network (i.e., according to the sequence diagram described in clause 8.3.1).

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 the data handling framework

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

Figure I.1 gives an example of realization of the data handling framework described in this Recommendation on an IMT-2020 network [b-ITU-T Y.3104] and [ITU-T Y.3111].

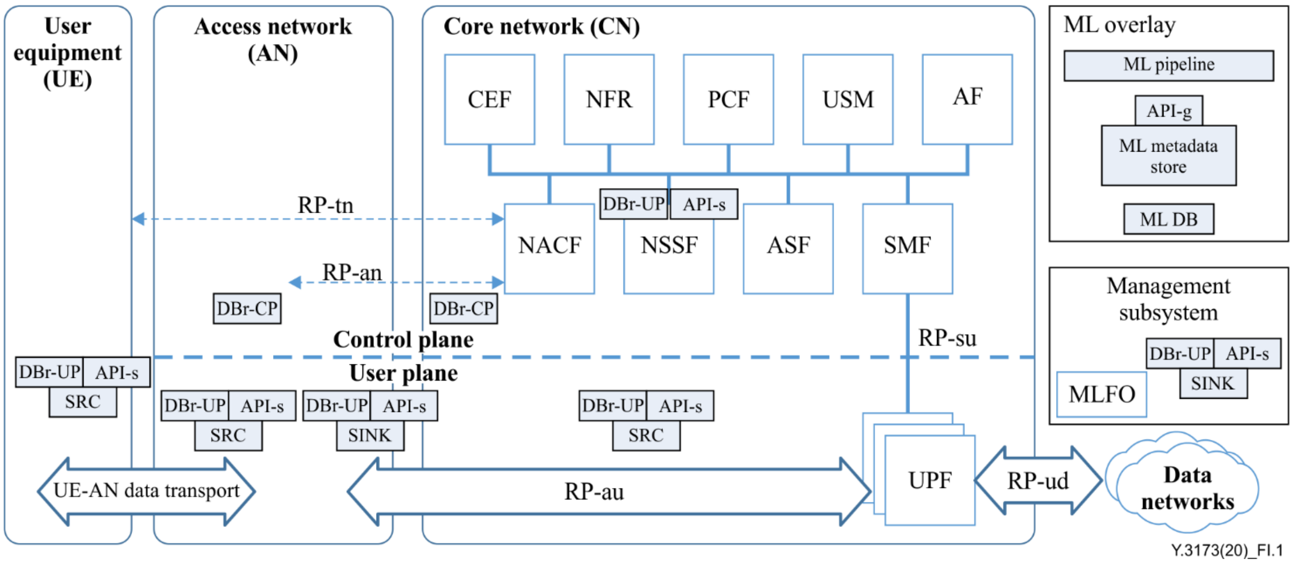


Figure I.1 – Example of realization in an IMT-2020 network

The example of realization is represented in the following manner: The positions of the ML database (ML DB), ML data broker components (i.e., DBr-CP and DBr-UP), ML metadata store and API-s and API-g are illustrated along with the underlay network.

Taking the load balance and cell splitting/merging use case as an example, the SRC, DBr-UP and DBr-CP can be deployed in the radio access network, the ML metadata store and the ML DB in the ML overlay and the MLFO in the management subsystem. To facilitate the ML learning enabled load balance and cell splitting and merging, load prediction and mobility prediction are needed. The well-defined DMs 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 DMs exist. If yes, the MLFO creates a data broker session for this ML application. After the MLFO has instantiated the DBr-UP and DBr-CP, the SRC node collects data from the radio access network via API-s and the corresponding data are provided to the ML pipeline deployed in the ML overlay via API-g. The ML data output for these ML applications, e.g., cell reselection parameters and/or handover parameters, are delivered to the SINK nodes in the ML pipeline via API-g and then to radio access networks 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.)

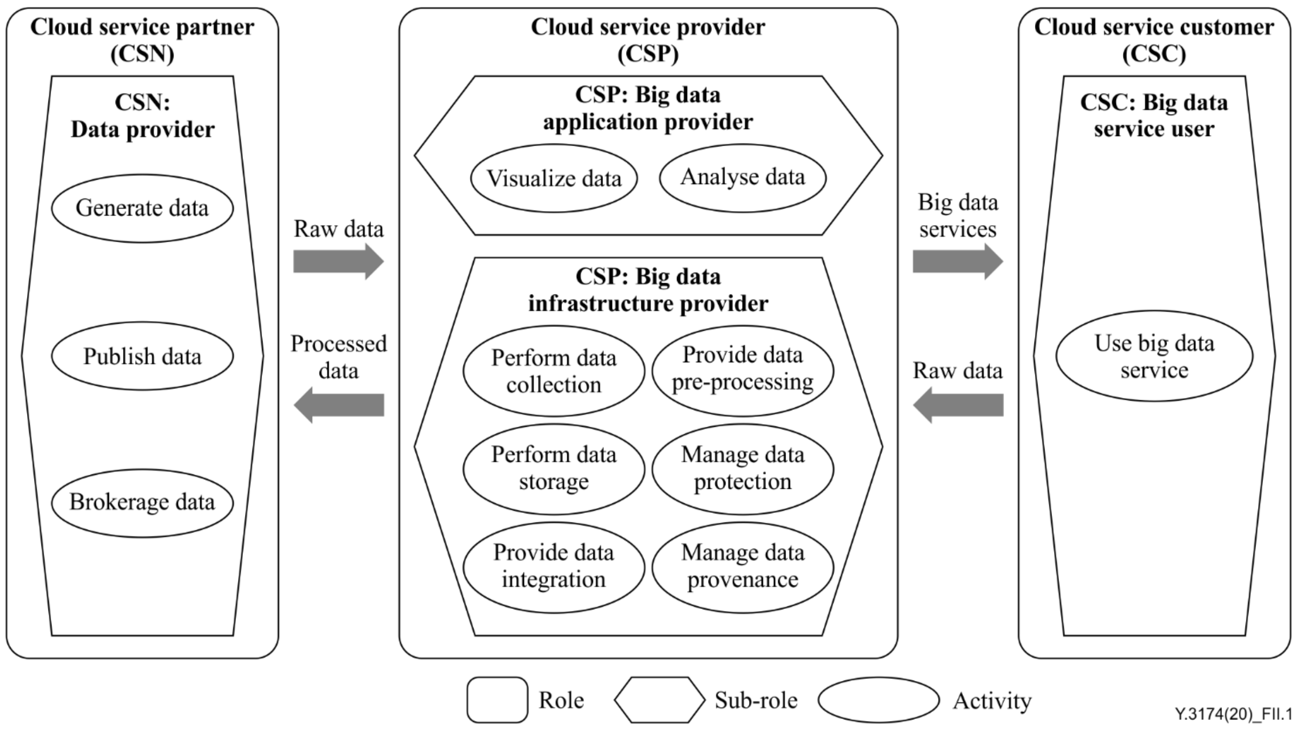
The roles in cloud computing based big data system context are defined in [b-ITU-T Y.3600] such as:

– cloud service partner (CSN):data provider (DP) (CSN:DP);

– cloud service provider (CSP):big data infrastructure provider (CSP:BDIP);

– CSP:big data application provider (CSP:BDAP);

– cloud service customer (CSC):big data service user (CSC:BDSU).



NOTE – Taken from Figure 7-1 in [b-ITU-T Y.3600]

Figure II.1 – Cloud computing based big data system context

[b-ITU-T Y.3600] also defines the activities for the above listed roles. The data provider (CSN:DP) includes data generation, data publishing and data brokerage activities. The 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 the CSP.

Table I.5 – Mobile network user behaviour big data analysis in [b-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 a private big data infrastructure using cloud computing technologies or use services provided by a cloud service provider: big data infrastructure provider (CSP:BDIP) and build big data application services or use services provided by a cloud service provider: big data application provider (CSP:BDAP). Telecom operators could also act as a 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 support the roles and activities of CSN:DP, CSP:BDIP, CSP:BDAP and 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 [b-ITU-T Y.3600]. Thus, the telecom operator is able to use a uniform architecture in case of deployments of ML pipeline in future networks including IMT-2020, as well as cloud computing based big data analysis.

Figure II.2 provides an example of instantiation of data handling framework for ML in a cloud computing based big data environment under agreement between a CSP and a telecom operator. In such deployments, reference point 4 could use the data handling framework described in this Recommendation.

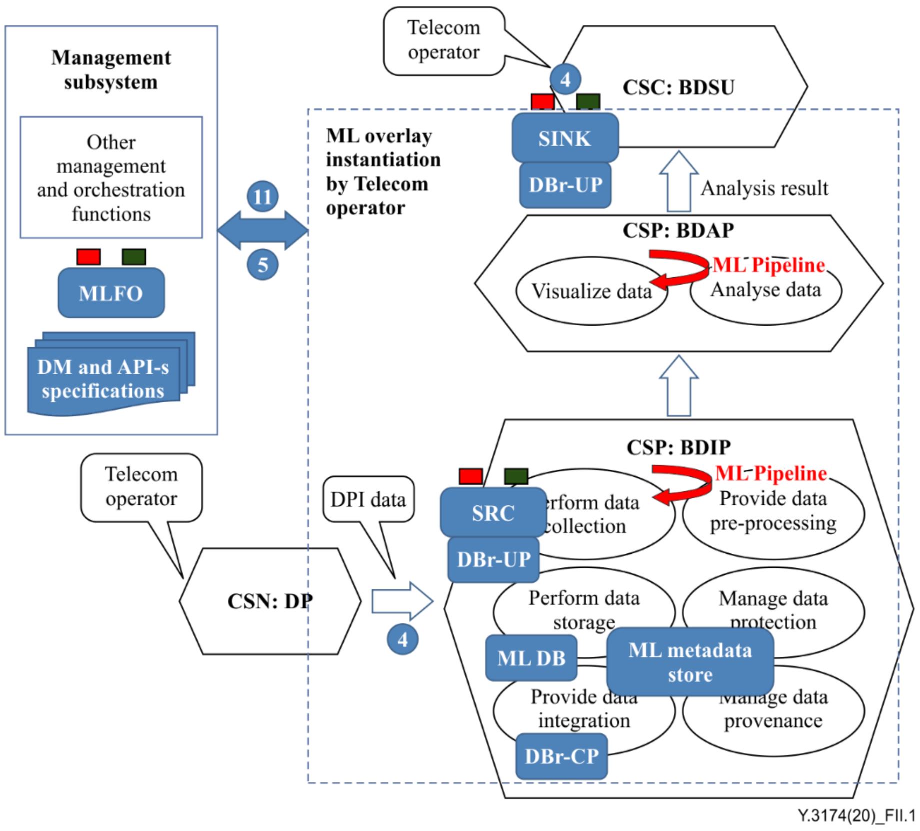


Figure II.2 – Example of instantiation of data handling framework for ML in cloud computing based big data

Table II.1 provides a mapping of functionalities between the data handling framework for ML described in this Recommendation and the services provided by cloud computing based big data.

NOTE – The mapping assumes that telecom operator buys cloud services from CSP and builds and/or customizes 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 network	For the purposes of ML applications, telecom operators' networks would function as ML underlay networks. The activities of CSN:DP and CSC:BDSU are mapped to the network functions that correspond to SRC nodes and SINK nodes for ML output, respectively.
DBr-UP	The activities of the CSP:BDAP and CSP:BDIP can be mapped to these functionalities, 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 as 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, reference point 5 [ITU-T Y.3172] and reference point 11	It is expected that the service orchestrator in the CSP:BDAP can support the MLFO functionalities and can use reference points 5 and 11 to manage the ML overlay components and data handling components, respectively.
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 the transfer of raw data, processed data and big data services, are mapped to the reference point 4.

Bibliography

[b-ITU-T Y.3100]	Recommendation ITU-T Y.3100 (2017), *Terms and definitions for IMT-2020 network*.
[b-ITU-T Y.3104]	Recommendation ITU-T Y.3104 (2018), *Architecture of the IMT-2020 network*.
[b-ITU-T Y.3600]	Recommendation ITU-T Y.3600 (2015), *Big Data-Cloud computing based requirements and capabilities*.
[b-ITU-T Y.Sup.55]	Supplement 55 to ITU-T Y-series Recommendations (2019), *ITU-T Y.3170-series – Machine learning in future networks including IMT-2020: Use cases*.
[b-ETSI 129 500]	ETSI TS 129 500 V15.0.0 (2018-07) 5G; *5G System; Technical Realization of Service Based Architecture*.
[b-ETSI GR ENI 004]	ETSI GR ENI 004 V1.1.1 (2018), *Experiential Networked Intelligence (ENI); Terminology for Main Concepts in ENI*.
[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-3GPP 38201]	3rd Generation Partnership Project; *Technical Specification Group Radio Access Network*; *NR; Physical layer; General description*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2020