

## **Draft Recommendation ITU-T Y.3172 (formerly Y.IMT2020-ML-Arch)**

### **Architectural framework for machine learning in future networks including IMT-2020**

#### **Summary**

This document specifies an architectural framework for machine learning (ML) in future networks including IMT-2020. A set of architectural requirements and specific architectural components needed to satisfy these requirements are presented. These components include, but are not limited to, ML pipeline as well as ML management and orchestration functionalities. The integration of such components into future networks including IMT-2020 and guidelines for applying this architectural framework in a variety of technology-specific underlying networks are also described.

#### **Keywords**

architectural framework, IMT-2020, high-level architecture, intent, machine learning, ML, orchestrator, overlay, pipeline, requirements, sandbox, training, underlay

## Table of Contents

1	Scope .....	3
2	References .....	3
3	Definitions .....	3
3.1	Terms defined elsewhere.....	3
3.2	Terms defined in this Recommendation .....	3
4	Abbreviations and acronyms .....	4
5	Conventions.....	5
6	Introduction .....	6
7	High-level architectural requirements .....	7
7.1	Enablers for correlation .....	7
7.2	Enablers for deployment .....	9
7.3	Interface-related .....	11
7.4	Declarative specification .....	12
7.5	Management of ML functionalities .....	14
8	Framework of the high-level architecture .....	16
8.1	High-level architectural components .....	16
8.2	High-level architecture .....	18
8.3	General guidelines for realization of the high-level architecture.....	21
9.	Security considerations .....	21
	Appendix I.....	22
	Appendix II .....	24
	Bibliography.....	26

## Draft Recommendation ITU-T Y.IMT2020-ML-Arch

### Architectural framework for machine learning in future networks including IMT-2020

#### 1 Scope

This Recommendation specifies an architectural framework for machine learning in future networks including IMT-2020.

A set of architectural requirements is presented, which in turn leads to specific architectural components needed to satisfy these requirements. This Recommendation also describes an architectural framework for integration of such components into future networks including IMT-2020 and guidelines for applying this architectural framework in a variety of technology-specific underlying networks.

#### 2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

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

#### 3 Definitions

##### 3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1 Network function [b-ITU-T Y.3100]:** In the context of IMT-2020, a processing function in a network.

NOTE 1 – Network functions include but are not limited to network node functionalities, e.g., session management, mobility management and transport functions, whose functional behaviour and interfaces are defined.

NOTE 2 – Network functions can be implemented on a dedicated hardware or as virtualized software functions.

NOTE 3 – Network functions are not regarded as resources, but rather any network functions can be instantiated using the resources.

##### 3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1 machine learning (ML):** 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 two examples of machine learning types.

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

**3.2.3 machine learning model:** model created by applying machine learning techniques to data to learn from.

NOTE 1 – A machine learning model is used to generate predictions (e.g. regression, classification, clustering) on new (untrained) data.

NOTE 2 – A machine learning model may be encapsulated in a deployable fashion in the form of a software (e.g. virtual machine, container) or hardware component (e.g. IoT device).

NOTE 3 – Machine learning techniques include learning algorithms (e.g. learning the function that maps input data attributes to output data).

**3.2.4 machine learning overlay:** a loosely coupled deployment model of machine learning functionalities whose integration and management with network functions, are standardised.

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

**3.2.5 machine learning pipeline:** 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].

**3.2.6 machine learning sandbox:** 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.

**3.2.7 machine learning underlay network:** a telecommunication network and its related network functions which interfaces with corresponding machine learning overlays.

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

## 4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AF	Application Function
AN	Access Network
API	Application Programming Interface
AR/VR	Augmented Reality/Virtual Reality
C	Collector (in ML pipeline)
CN	Core Network
D	Distributor (in ML pipeline)
EMS	Element Management System
FMC	Fixed Mobile Convergence

GPU	Graphic Processor Unit
IoT	Internet of Things
KPI	Key Performance Indicator
M	Model (in ML pipeline)
mIoT	massive Internet of Things
MEC	Multi-access Edge Computing
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
MPP	Mobility Pattern Prediction
MnS	Management Service
NF	Network Function
NOP	Network Operator
OAM	Operations, Administration and Maintenance
P	Policy (in ML pipeline)
PP	Pre-Processor (in ML pipeline)
QoS	Quality of Service
RCA	Root Cause Analysis
RRC	Radio Resource Control
SBA	Service-Based Architecture
SMF	Session Management Function
SON	Self-Optimizing Network
SRC	Source
UE	User Equipment
UPF	User Plane Function
V2X	Vehicle-to-everything
VoLTE	Voice over Long-Term Evolution

## 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 Recommendation is to be claimed.

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

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

In the body of this Recommendation and its annexes, the words shall, shall not, should, and

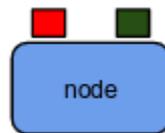
may sometimes appear, in which case they are to be interpreted, respectively, as is required to, is prohibited from, is recommended, and can optionally. The appearance of such phrases or keywords in an appendix or in material explicitly marked as informative is to be interpreted as having no normative intent.

**ML pipeline** – In this Recommendation, when the symbol shown in Figure 1 is used, this denotes a subset (including proper subset) of nodes in an ML pipeline. When this symbol is used in a figure, the symbol stands for the subset of an ML pipeline’s nodes not explicitly shown in that figure.



**Figure 1 - Symbol used to denote a subset of nodes in an ML pipeline**

**Service egress and ingress points** – In this Recommendation, a node and its service egress and ingress points are denoted by the symbol in Figure 2. The service egress point is shown with a green rectangle on top of the node, while the service ingress point corresponds to the red rectangle on top of the node.



**Figure 2 - Symbol used to denote a node with its service egress and ingress points**

## 6 Introduction

Machine learning (ML) provides a way to teach computational systems to gain knowledge from data without necessarily being explicitly programmed in order to realize complicated tasks such as detection of characteristics or prediction of behaviours. As ML becomes an important technical trend in the industry, network operators and other stakeholders are searching for cost-effective ways to incorporate ML into the future networks including IMT-2020.

While the benefits from such an integration have been discussed under many use cases (e.g., troubleshooting of network problems, network traffic prediction, traffic optimization adjustment, network security auditing [b-ITU-T Y.3650] [b-ITU-T Y.3170]), there are many challenges to such an integration. Some of the important challenges are:

- A. Heterogeneous nature of ML functionalities and unique characteristics of future communication technologies impose a varied set of requirements for integration.
- B. Roadmaps for evolution of these ML functionalities and communication networks are not aligned.
- C. Cost of integration, in terms of architecture impacts, is an important consideration.
- D. Disparate management mechanisms for ML functionalities and network functions [b-ITU-T Y.3100] will disrupt the operations management of communication networks.

An architectural framework for integration of ML with future networks including IMT-2020 is provided to address these challenges.

Building on high-level architectural requirements, such a framework at first provides a common vocabulary and nomenclature for ML functionalities and their relationships with the communication networks. The framework addresses the reference points for future networks including IMT- 2020 which enable loosely coupled integration with ML functionalities. Management mechanisms for ML are also described in the framework considering the existing management principles which have been identified in IMT-2020 networks.

The framework enables a standard method of integrating ML functionalities in future networks including IMT-2020.

## 7 High-level architectural requirements

This clause provides high-level requirements for the design of the high-level architecture framework in clause 8.

These high-level architectural requirements are classified as follows:

- Enablers for correlation of data across levels and heterogenous technologies. This set of requirements addresses the challenge A mentioned in clause 6 and the requirements which fall into this set are provided in clause 7.1.

NOTE – Levels may correspond to different parts of the network where the reference points between these parts are defined in the relevant network architecture specifications. For example, [b-ITU-T Y.3104] defines the reference point RP-an between Access Network (AN) and Core Network (CN), thus, AN and CN may correspond to different levels of an IMT-2020 network.

- Enablers for deployment. This set of requirements addresses the challenge B mentioned in clause 6 and the requirements which fall into this set are provided in clause 7.2.
- Requirements related to interfaces between the architectural components. This set of requirements addresses the challenges B and C mentioned in clause 6 and the requirements which fall into this set are provided in clause 7.3.
- Requirements related to declarative specifications used for specifying the ML applications. This set of requirements addresses the challenges C and D mentioned in clause 6 and the requirements which fall into this set are provided in clause 7.4.
- Requirements related to the management of the architectural components. This set of requirements addresses the challenge D mentioned in clause 6 and the requirements which fall into this set are provided in clause 7.5.

### 7.1 Enablers for correlation

Table 7-1 provides requirements regarding enablers for correlation of data across levels and heterogeneous technologies.

**Table 7.1 – High-level requirements – Enablers for correlation**

REQ-ML-COR-001	The ML architecture is recommended to support correlation of data coming from multiple sources.
Description	In future networks, sources of data may be heterogeneous, integrated with different network functions (NFs), and may report different formats of data. These varied "perspectives" can provide rich insights upon correlated analysis. Architectural components to enable the ML functionalities to collect and correlate data from these varied sources in the network are needed.
Notes	NOTE 1 – As an example, the analysis of data from user equipment (UE), AN, CN and application function (AF) is needed to predict potential issues related to quality of service (QoS) in end-to-end user flows. NOTE 2 – Other examples of such sources of data are self-optimizing network (SON) functionalities that monitor and correlate network alarms

	and key performance indicators (KPIs) [b-3GPP TS 28.554], and then take relevant action to clear alarms or enhance network KPIs, or give network design recommendations without human intervention.
--	---

REQ-ML-COR-002	<p>a) The ML architecture is required to support multiple technologies of future networks including IMT-2020 to achieve end-to-end user experience.</p> <p>b) The ML architecture is recommended to be interfaced with non-IMT-2020 external functional entities to achieve end-to-end user experience.</p>
Description	<p>Future networks will have multiple technologies coexisting side by side, e.g., licensed and unlicensed wireless technologies, fixed mobile convergence (FMC) technologies, legacy and future technologies. The emergence of network slicing [ITU-T Y.3111] is one example in which vertical technologies (networks customized to provide flexible solutions for different market scenarios) and their integration into future networks are important. The interfacing of ML architecture with such functional entities will help in achieving KPIs as for example specified in [b-3GPP TS 28.554].</p> <p>Thus, it is important for the ML architecture to be capable of integration with multiple underlying technologies and even support of application functions.</p>
Notes	<p>NOTE 1 – Vehicle-to-everything (V2X) is an example of a vertical application which may benefit from the support of network slicing.</p> <p>NOTE 2 – Various communication network (e.g. 3G, 4G, 5G) technologies could be considered as examples of underlying technologies.</p> <p>Applications for hosting in-car entertainment, processing data from drones, entertainment applications using augmented reality/virtual reality (AR/VR) are examples of application functions.</p> <p>NOTE 3 – Examples of network functions which are not directly managed by the network operator (NOP) are sensors, power circuits and different Internet of things (IoT) modules. In some use cases, the data from such network functions are utilized to control and monitor both the network functions deployed in the network as well as such external network functional entities themselves. These data could also be then used in various types of network parameter optimizations to achieve gains in coverage, capacity and quality by the NOP.</p>

REQ-ML-COR-003	The ML architecture is required to support distributed instantiation of machine learning functionalities in multiple levels.
Description	<p>Distributed instantiation helps in using data from different levels, to enrich locally available data in a level, as needed. Thus, the ML functionalities may be multi-level.</p> <p>Independent instances of ML functionalities may be created in multiple levels.</p>
Notes	NOTE 1 – In some use cases, the source of data may be placed in AN and pre-processed data are handled by CN [b-ITU-T Y.3102] where the ML model is hosted.

	NOTE 2 – ML functionalities may be used for performing mobile pattern predictions (MPPs) or network slice [b-ITU-T Y.3100] configurations using input from the network at multiple levels (from sources in AN or CN).
--	---

## 7.2 Enablers for deployment

Table 7-2 provides requirements regarding enablers for deployment of ML functionalities.

**Table 7-2 – High-level requirements – Enablers for deployment**

REQ-ML-DEP-001	The ML architecture is recommended to define the points of interaction between ML functions and technology-specific underlay network functions, independently from functionalities of the machine learning application.
Description	The specification of ML applications in future networks will be related to the existing (or new) network services or network functions. Based on the specification of the ML applications, placement and characteristics of the ML functionalities are decided. The source of data and the target of ML output are points of tight integration with the technology-specific underlay NFs. Apart from this, ML functionalities may be generic and do not have tight integration with technology-specific NFs. Well-defined reference points between the ML functionalities and the technology-specific underlay NFs are required.
Notes	NOTE 1 – ML output may be policies or configurations to be applied in the network and target of ML output may be functions in the network for applying ML output. Such application of ML output may be controlled by network operator policies. NOTE 2 – In some use cases, source of data and traffic classification (based on ML output) may be placed in the user plane of the network, e.g. user plane function (UPF) [b-ITU-T Y.3104]. These may be considered as points of tight integration between the ML functionalities and the underlying network (e.g. AN or CN). Other ML functionalities (e.g., the ML model) do not have such interface dependencies on the underlying networks.

REQ-ML-DEP-002	The ML architecture is required to support split or combined deployments of ML functionalities across different underlay network functions.
Description	In future networks, management and orchestration functions [ITU-T Y.3111] will optimize the location and the performance of NFs accordingly. To carry forward such benefits to ML applications, similar optimizations should also be applied to ML functionalities. Moreover, the constraints applicable to an ML functionality may be unique.
Notes	NOTE 1 – ML training may need a graphic processor unit (GPU) and may need to be done in an isolated environment so that it does not affect other functionalities of the network.

	NOTE 2 – As an example, depending on the latency budget, data availability and other considerations for ML applications, the ML functionalities could have the source of data and the ML training hosted in the CN or AN.
--	---

REQ-ML-DEP-003	The ML architecture is required to support flexible placement of ML functionalities (in coordination with the management and orchestration functions [b-ITU-T Y.3100] [b-ITU-T Y.3110]) in the underlying network functions.
Description	<p>The flexible placement ML functionalities could be based, among other factors, on the specifications of ML applications.</p> <p>The Management and Orchestration functions utilize both the specifications of the ML applications and the conditions in the network to implement this requirement.</p> <p>Resource allocation for ML functionalities is required to consider various constraints (e.g., resource constraints of the NFs, latency constraints specific to the ML application, availability of data that is specific to the ML application, data transformation capabilities, performance constraints, training capabilities and model characteristics).</p> <p>The ML architecture should provide the ability to place the ML functionalities in a flexible manner in the network that is most optimal for the performance of the ML applications and based on the constraints defined in the declarative specifications of the ML applications.</p> <p>The constraints for online training and prediction for real-time ML applications which are captured in the specification form inputs to placing the ML functionalities in the network that can provide optimal performance for the use case.</p>
Notes	<p>NOTE 1 – If the ML model includes a neural network, then a placement decision in a GPU-based system is desirable.</p> <p>NOTE 2 – Based on the requirements of the ML applications, the ML functionalities which provide latency sensitive short-term predictions may be hosted closer to the edge. The placement may also be influenced by considerations on data availability. This may be done in coordination with the split of ML functionalities mentioned in REQ-ML-DEP-002.</p> <p>NOTE 3 – In certain use cases, user plane data classification may be done using ML models. Since user plane data classification is a latency-sensitive application, the model may be hosted at the transport network, whereas the training could be done at CN.</p>

REQ-ML-DEP-004	The ML architecture is required to support plugging in and out new data sources or configuration targets to a running ML environment.
Description	Certain advanced applications in future networks, e.g., massive Internet of Things (mIoT), require handling of unstructured data from a huge number of data sources that may be plug and play. One such use case is the analysis of logged data for anomaly detection in networks.
Notes	<p>NOTE 1 – Scaling of ML functionalities based on type and volume of incoming data is an example of handling new data sources.</p> <p>NOTE 2 - The configuration of ML functionalities in the network needs to</p>

	handle the plugging in and out of new data sources or configuration targets based on metadata of data sources.
--	--

### 7.3 Interface-related

Table 7-3 provides interface-related requirements.

**Table 7-3 – High-level requirements – Interface-related**

REQ-ML-INT-001	The ML architecture is recommended to support an interface to transfer trained models among ML functionalities of multiple levels.
Description	Training of models has certain specific needs, e.g., availability of certain kinds of processors, availability of certain kinds of training data. Once the training is done, the ML model has to be sent to the technology-specific underlay network that is hosting the ML model. Model training can be done separately from the live network. Thus, sending trained models across multiple levels is an important requirement.
Notes	NOTE 1 – UE, access network functions (AN), core network (CN) functions could be treated as examples of technology-specific underlying network functions which could host the trained model. NOTE 2 – CN and AN could be treated as examples of multiple levels. NOTE 3 – Depending on the availability of data, learning may be done at the CN or AN, and trained model (e.g. classifier) hosted by the transport networks.

REQ-ML-INT-002	The ML architecture is required to support an interface to transfer data for training or testing models among ML functionalities of multiple levels.
Description	Certain levels in which the data are available may not have the training capabilities. In such cases, there may be a need to send data for training or testing to levels where the capacity for such operations is available.
Notes	NOTE 1 – ANs may be considered as examples of resource-constrained networks, whereas CNs may be considered as a level where capacity may be available to scale. NOTE 2 – Data pre-processing may be done at the transport network and the pre-processed data is then sent to the model at the CN for training.

REQ-ML-INT-003	<ul style="list-style-type: none"> <li>a) While defining the interface with underlying networks, the ML architecture is recommended to utilize existing standard protocols wherever possible, with required extensions wherever needed.</li> <li>b) The ML architecture is recommended to support specific interfaces or application programming interface (API) for interfacing with technology-specific network functions, for sourcing data from such NFs or for configuring such NFs.</li> <li>c) The ML architecture is recommended to support logical interfaces between ML functionalities which can be hosted in multiple levels, and the realization of such logical interfaces be implemented according to deployment scenarios.</li> </ul>
Description	Sources of data and target for ML output may need specific interfaces or

	<p>APIs with the underlay networks to extract data or apply configurations. Some use cases may need a tight coupling at integration stage between the source and target of ML functionalities, and the NFs. In certain cases, an extension of such interfaces may be needed to achieve the ML functionalities in the use case.</p> <p>The ML functionalities may use interfaces provided by an underlying network as a source of data or target of configurations. In that sense, these network-specific APIs may act as realizations of an interface to the source and target of ML functionalities.</p>
Notes	<p>NOTE 1 – With respect to item a above, certain interfaces may be realized by reusing existing protocols (e.g., Radio Resource Control (RRC) [b-ITU-T Y.3104], MEC [b-ETSI MEC 003], management service (MnS) [b-3GPP TS 23.501]).</p> <p>NOTE 2 – With respect to item b above, for example, a source running in the UE may use specific APIs to extract data from a voice over long-term evolution (VoLTE) client [b-GSMA IR.92].</p> <p>NOTE 3 – With respect to item c above, for example, in the use case where the source runs in the AN but needs measurements from the UE, the AN needs to configure the UE for this measurement using RRC.</p>

REQ-ML-INT-004	The ML architecture is recommended to support data sharing between ML functionalities using distributed data sharing mechanisms.
Description	Cross-level sharing of data is needed to enable correlated ML decisions in future networks.
Notes	NOTE - Concepts like data lakes are emerging in future clouds and can also be exploited in network operators' clouds.

#### 7.4 Declarative specifications

Table 7-4 provides requirements regarding declarative specifications.

**Table 7-4 – High-level requirements – Declarative specification**

REQ-ML-SPEC-001	The ML architecture is required to support a standard method to represent ML applications, which can be translated into ML functionalities in technology-specific underlay network functions.
Description	Automation using declarative specification and corresponding translation into configurations is a characteristic of future networks. Extending this technique to ML, declarative specification of ML applications and corresponding translation into configurations of ML functionalities need to be supported.
Notes	Interpretation of the declarative specification allows configuration of ML functionalities in the network by translating the specification into the configuration that can be hosted by network functions.
REQ-ML-SPEC-002	<p>a) The ML architecture is required to support ML applications to specify the sources of data, repositories of ML models, targets for output from ML models, and constraints on network resources.</p> <p>b) The ML architecture is required to support the time constraints</p>

	requirements of ML applications.
Description	<p>The separation between technology agnostic part of the ML application and technology-specific deployment is captured in the design time of future network services. Declarative specifications for the ML applications achieve this separation.</p> <p>Different ML applications have varied time constraints. These constraints form an important input to the management and orchestration functions while determining the placement, chaining and monitoring of the ML functionalities.</p>
Notes	<p>NOTE 1 – With respect to item a) above, as an example, a description written in a metalanguage may capture the requirements of a network operator for an ML application. The management and orchestration functions may translate it into the configuration that can be implemented in the network.</p> <p>NOTE 2 – With respect to item b) above, at the tightest scale, the application of ML in beamforming, scheduling, link adaptation network functionalities would have latency criteria of the order of microseconds, whereas transport and core network functionalities have a few milliseconds of latency criteria. The least demanding in terms of latency are management level functionalities, e.g., anomaly detection and coverage hole detection, that can afford minutes, hours or days of latency.</p>

REQ-ML-SPEC-003	The ML architecture is required to support flexible split of ML functionalities based on the specifications of ML applications.
Description	<p>Specification of ML application is an important input for deployment of ML in future networks including IMT-2020. But network capabilities can change (hardware can be added or removed), NFs may be scheduled or (re)configured dynamically by the management and orchestration functions. These dynamic changes may necessitate a change in the split and placement of the ML functionalities (e.g., a decision may be taken to collocate certain functions of ML, based on changes in the link capacity, or a decision may be taken to add a new source of data based on decisions of the management and orchestration functions). Thus, a combination of (a) inputs from specification, (b) the requirement of ML functions to be capable of split and combined deployment and (c) coordination with the underlying management and orchestration function, is needed.</p>
Notes	NOTE – A new source of data may be instantiated based on scale out decisions in the network.

REQ-ML-SPEC-004	The ML architecture is required to support the specifications of ML applications by third parties to specify the sources of data, repositories of ML models, targets for output from ML models, and constraints on network resources.
-----------------	---

Description	Third party [b-ITU-T Y.3100] service providers may offer innovative services on top of future networks. This may include new ML algorithms. Collaboration between third party providers and network operators may bring new sources of data or aggregation capabilities. The declarative process in the architecture should extend the capabilities to include third parties, and they should be able to include these functionalities in the specification so that end users can benefit from such innovative services offered by third party providers.
Notes	<p>NOTE 1 – In some use cases, ML models may be provided as third-party applications.</p> <p>NOTE 2 – A smartphone application which interfaces with the sensors on the UE is an example of a third-party source of data.</p> <p>NOTE 3 – A third-party voice over IP service provider wants to optimize call quality over the network by running an ML application that configures network parameters. Specification of such an ML application is an example of a third-party specification.</p>

## 7.5 Management of ML functionalities

Table 7-5 provides requirements regarding the management of ML functionalities.

**Table 7-5 – High-level requirements – Management of ML functionalities**

REQ-ML-MNG-001	The ML architecture is required to support ML model selection at the setup time of the ML functionalities.
Description	<p>Advances in machine learning suggest that in future networks there would be ML models with varied characteristics (e.g., using a variety of optimization techniques and weights) that are appropriate for different problem spaces and data characteristics.</p> <p>In future networks, new sources of data may get added dynamically. To extend the ML applications to such new and heterogeneous sources of data, ML model selection has to be done dynamically, based on the data provided by the sources.</p>
Notes	NOTE– Plug and play of new network functions (e.g. new UPF) into a live network may be an example for dynamic on-boarding of new sources of data.
REQ-ML-MNG-002	The ML architecture is required to support ML model training and model updates while preventing impact on the network.
Description	<p>ML model training has several considerations: use of specific hardware for speed, availability of data, parameter optimizations, avoiding bias, distribution of training, etc.</p> <p>Moreover, in future networks, service disruptions should be avoided while model training and updates are performed.</p>
Notes	

REQ-ML-MNG-003	The ML architecture is required to support capabilities to monitor the network operations based on the effect of ML and to update the ML models and/or policies without impacting the network.
Description	The effect of ML on the network needs to be monitored. Various KPIs are measured constantly and the impact of machine learning on them as well as on the ML functionalities themselves needs to be monitored and corrected if needed, continuously. ML functionalities need to be trained for future recognition and handling such corrected scenarios.
Notes	NOTE– Continuous improvement of the automated fault recovery process workflows is an example of a process where such monitoring and update is done in the network. The root cause analysis (RCA) from the fault recovery process is provided for configuring the management and orchestration functions, and also the effect produced by ML in the network is evaluated and used to optimize the ML functionalities themselves.

REQ-ML-MNG-004	The ML architecture is required to support an orchestration functionality to manage all the ML functionalities in the network.
Description	<p>The performance of the ML functionalities in the network is monitored and when the performance falls below a predefined threshold, the ML functionalities are reconfigured to improve it.</p> <p>The varied sources of data (e.g. CN, AN) imply that there could be various training techniques including distributed training. Complex models may be trained using varied data. The performance of such models can be determined and compared in an isolated environment.</p> <p>Based on such comparisons, network operators can then select the appropriate ML functionalities (based on internal policies) for specific ML applications.</p>
Notes	<p>NOTE 1 – Evaluation involves evaluation of network performance along with performance of ML algorithms.</p> <p>NOTE 2 – Reselection of ML model is an example of reconfiguration done to improve the performance of the ML functionalities.</p>

REQ-ML-MNG-005	The ML architecture is required to support flexible chaining of ML functionalities including multi-level chaining.
Description	<p>Flexible chaining of ML functionalities is required to be done based on the hosting and positioning on different NFs and levels. This is to enable the hybrid or distributed ML functionalities.</p> <p>Chaining of ML functionalities may be used to build a complex ML functionality.</p> <p>Management and orchestration functions provide NOPs with capabilities to rapidly design, develop and deploy network services in the technology-specific underlay networks. Similarly, ML functionalities in the network need mechanisms including flexible chaining to keep up with innovation in the technology-specific underlay networks. As underlying network services evolve and deploy rapidly, so do the ML functionalities on top of them, using flexible chaining techniques. This requirement aims to give the ML functionalities, the ability to adapt to dynamic service creation and</p>

	orchestration in the underlying networks.
Notes	

## 8 Framework of the high-level architecture

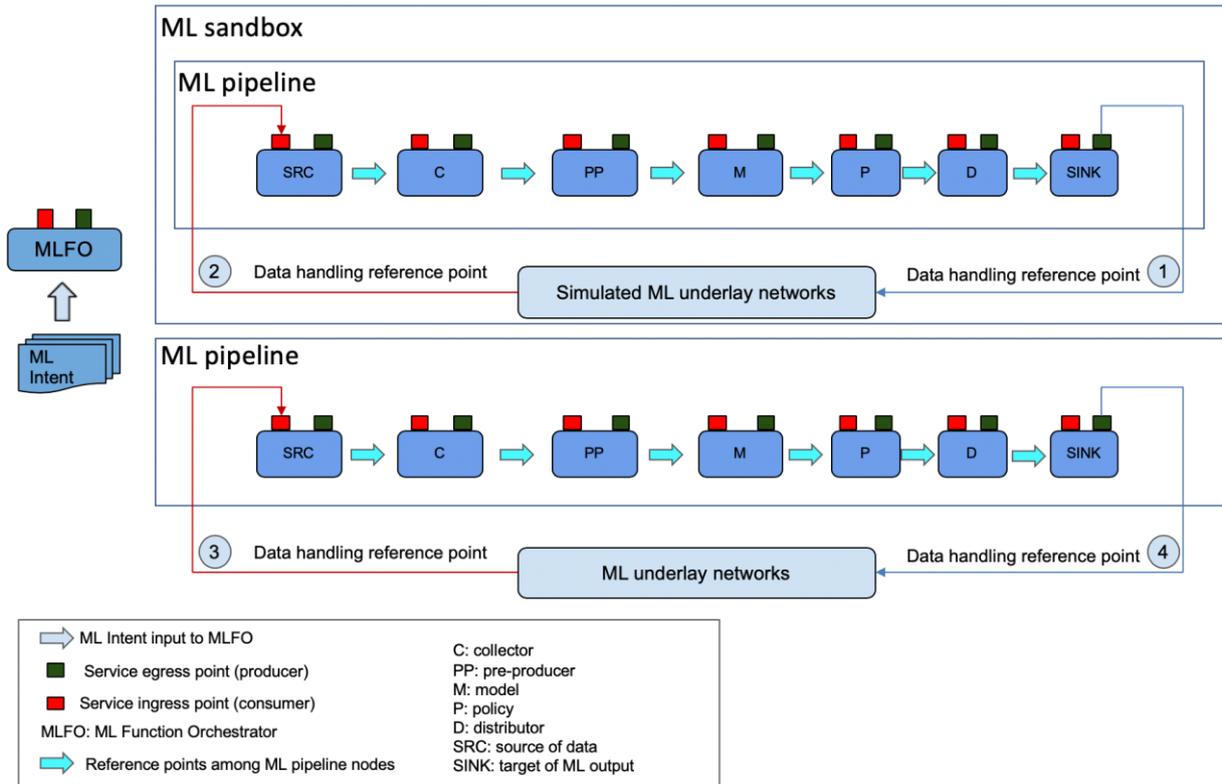
The framework of the high-level architecture described in this clause supports the requirements defined in clause 7.

This clause provides description of the high-level architectural components and the high-level architecture itself, and provides guidelines for realisation of the high-level architecture on different technology-specific underlay networks.

This approach to design the architecture provides an ability to analyse both ML solutions which are agnostic to technology specific underlying networks and issues specific to integration of ML to such underlay networks. An example of the realization of this architectural framework on technology-specific underlay networks is provided in Appendix I.

### 8.1 High-level architectural components

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



**Figure 3 – High-level architectural components**

The high-level architectural components include the following:

#### 1) machine learning pipeline.

As defined in clause 3, 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.

NOTE 1 – Machine learning pipeline may be deployed on simulated or live ML underlay networks.

The following describes the nodes of an ML pipeline.

- SRC (source): This node is the source of data that can be used as input to the ML pipeline.  
NOTE 2 – Potential SRC nodes include user equipment (UE), session management function (SMF) [b-ITU-T Y.3104] and application function (AF) [b-ITU-T Y.3104].
- C (collector): This node is responsible for collecting data from one or more SRC nodes.  
NOTE 3 – A collector node may have the capability to configure SRC nodes. For example, the radio resource control (RRC) protocol [b-3GPP TS 23.501] can be used to configure user equipment (UE) acting as a SRC node. The collector node may use vendor specific operations, administration and maintenance (OAM) protocols to configure SMF acting as a SRC node. Such configurations may be used to control the nature of data, its granularity and periodicity while it is generated from the SRC.
- PP (pre-processor): This node is responsible for cleaning data, aggregating data or performing any other pre-processing needed for the data to be in a suitable form so that the ML model can consume it.
- M (model): This is a machine learning model, in a form which is usable in a machine learning pipeline.  
NOTE 4 – Example is an ML algorithm implemented in software as a NF [b-ITU-T Y.3104].
- P (policy): This node enables the application of policies to the output of the model node.  
NOTE 5 – This node can be used, for example, to minimize impacts when the output of machine learning is applied to a live ML underlay network. Specific rules can be put in place by a network operator to safeguard the sanity of the network, e.g., major upgrades may be done only at night time or when data traffic in the network is low.
- D (distributor): This node is responsible for identifying the SINK(s) and distributing the output of the M node to the corresponding SINK nodes.  
NOTE 6 – It may have the capability to configure SINK nodes. For example, RRC protocol may be used to configure a UE acting as a SINK node.
- SINK: This node is the target of the ML output, on which it takes action.  
NOTE 7 – For example, a UE acting as a SINK node may adjust the periodicity of channel measurement based on ML output.

NOTE 8 - A “service egress point” of a node is a point where services exposed by the node (producer role) can be accessed. A “service ingress point” of a node is a point from where services from other nodes can be consumed (consumer role).

## 2) machine learning function orchestrator (MLFO).

As defined in clause 3, MLFO is a logical node with functionalities that manage and orchestrate the nodes of ML pipelines based on ML Intent and/or dynamic network conditions.

NOTE 9 – MLFO selects and reselects the ML model based on, e.g. its performance.

NOTE 10 – The placement of the ML pipeline nodes, based on the corresponding network capabilities and constraints of the ML application, is the responsibility of the MLFO.

NOTE 11 – MLFO also provides chaining functionality, i.e., connecting ML nodes together to form an ML pipeline. For example, chaining can be used to connect an SRC node instantiated in the access network with C and PP nodes instantiated in the core network. The chain itself is declared in the ML application specification and its technology-specific implementation in the

network is done by the MLFO. The MLFO determines the chaining considering the constraints (e.g., timing constraints for prediction).

NOTE 12 – Figure 4 includes ML Intent to emphasise its important role as input to MLFO. ML Intent is a declarative description which is used to specify an ML application. ML Intent does not specify any technology-specific network functions to be used in the ML application and provides a basis for mapping ML applications to diverse technology-specific network functions. ML Intent may use a meta-language specific for machine learning to define ML applications.

### 3) ML sandbox.

ML sandbox is an isolated domain which allows hosting of separate ML pipelines to train, test and evaluate them before deploying them in a live network. For training or testing, ML sandbox can use data generated from simulated ML underlay networks and/or live networks.

NOTE 13 – ML sandbox may benefit from ML techniques such as supervised machine learning to train, test and evaluate ML models.

In addition to the above high-level architectural components, the following supporting **architectural aspects** are to be noted:

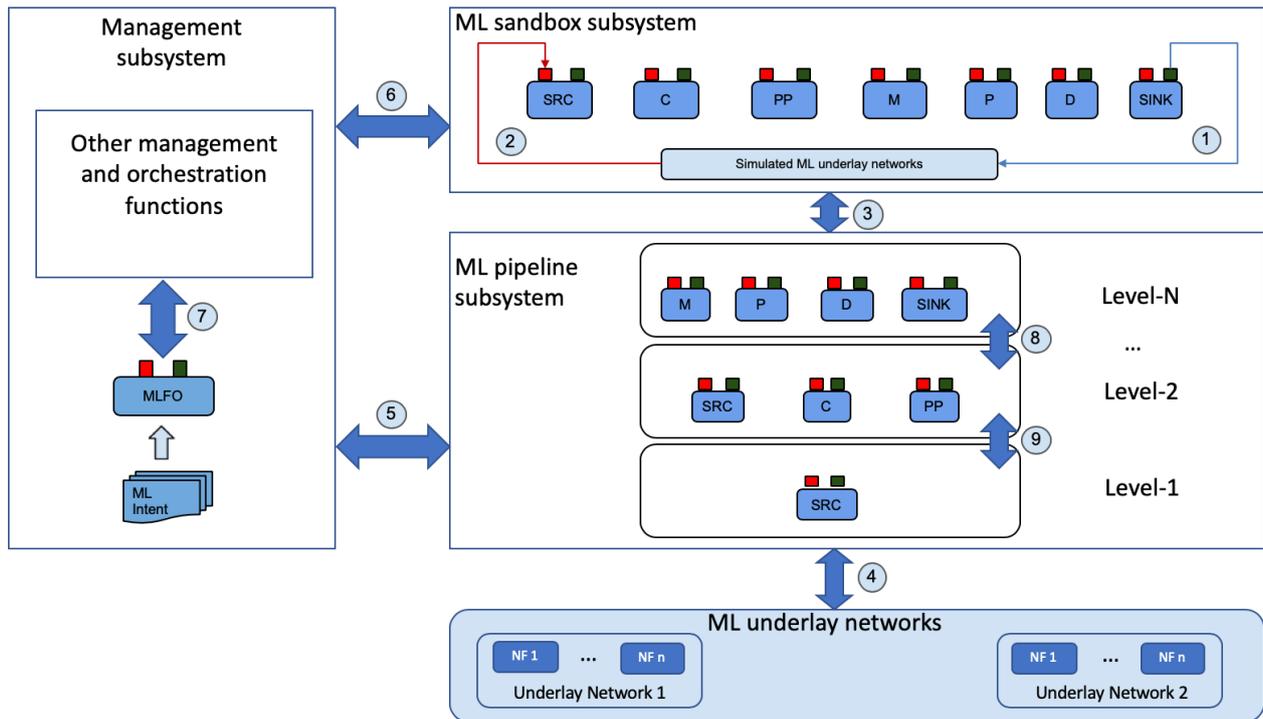
- Service-based architecture (SBA) [b-ETSI TS 129 500] may be used to interface ML functionalities with ML underlay networks. Similarly, for the ML pipeline in sandbox, SBA may be used to interface the ML functionalities with the simulated ML underlay networks. This provides a uniform reference point towards the ML overlay for NFs and the simulated ML underlay networks, alike. SBA may also be used between ML pipeline nodes themselves, and to manage the ML functionalities by the MLFO.
- Data handling reference points are defined between the ML pipeline and the simulated or live ML underlay networks. Using these reference points, the impacts to the ML underlay networks are localised to the source of data and target of configurations (as a result of ML pipeline execution).  
NOTE 14 – Extensions of existing protocols may be used at the data handling reference points to minimise the architectural impacts to the ML underlay networks.  
NOTE 15 – Non-SBA protocols need to be used at the data handling reference points in case the network functions in the ML underlay networks are not SBA-capable.

With reference to Figure 3, the arrows 2 and 3 show the paths to the ML pipeline for the data generated from the simulated ML underlay networks and the ML underlay networks, respectively. The arrows 1 and 4 show the paths from the ML pipeline for configuration of the target based on ML output in the simulated ML underlay networks and the ML underlay networks, respectively.

Appendix II provides the mapping of the architectural components and supporting architectural aspects to the high-level requirements provided in clause 7.

## 8.2 High-level architecture

The high-level architecture shown in Figure 4 is derived from the high-level requirements specified in clause 7 and builds upon the architectural components described in clause 8.1.



**Figure 4 – High-level architecture**

The **reference points** shown in Figure 4 are the following:

- 1, 2: Data handling reference points between simulated ML underlay networks and an ML pipeline in ML sandbox subsystem.
- 3: Reference point between ML sandbox subsystem and ML pipeline subsystem.
- 4: Reference point between ML pipeline subsystem and ML underlay networks.  
NOTE 1 – This reference point represents the data handling reference points shown as arrows 3 and 4 in Figure 3.
- 5, 6: Reference points between Management subsystem and, ML pipeline subsystem and ML sandbox subsystem, respectively.
- 7: Reference point between MLFO and other management and orchestration functions of Management subsystem.
- 8, 9: Reference points between ML pipeline nodes located in different levels.

The three subsystems of the high-level architecture as shown in Figure 4 are:

- **Management subsystem.** This subsystem includes MLFO and other management and orchestration functions, e.g. those defined in [ITU-T Y.3111].  
The management subsystem enables to extend the management and orchestration mechanisms used for future networks including IMT-2020 to ML pipeline nodes. This brings uniformity to the management of ML functionalities and NFs.  
The MLFO works in coordination with the other functions of the management subsystem to manage the ML pipeline nodes.  
NOTE 2 – The reference point(s) between the functions of the management subsystem and NFs in the ML underlay networks complies(y) with the reference points defined in the related specifications, e.g. [ITU-T Y.3111]. These reference points are out of scope of this Recommendation and thus are not shown in Figure 4.  
NOTE 3 – The interaction between MLFO and the other functions of the management subsystem may be achieved using service-based architecture (SBA).
- **ML pipeline subsystem.** The ML pipeline is a logical pipeline that can be overlaid on existing network infrastructures. The services of the MLFO are used for instantiation and setup.

Integration aspects of such ML pipeline overlay on technology-specific ML underlay networks may require extension or definition of specific protocols and APIs.

In addition, the following points are to be noted:

- SBA may be used for interfacing between NFs and ML pipeline nodes as well as between ML pipeline nodes themselves. As a first example, a SRC node exposes interfaces for consuming data from the NFs and producing data towards the collector (C) node. As a second example, a SINK node exposes interfaces for consuming the ML output from the distributor (D) node and produces such configurations to the NFs which it interfaces with.

NOTE 4 – Due to the heterogeneity of NFs and ML underlay networks, SBA may not be supported by NFs in the ML underlay networks. In such cases, protocols and APIs specific to those ML underlay networks are used between the ML pipeline nodes and the NFs.

- The placement and chaining of the ML pipeline nodes are controlled by the MLFO and this control may be influenced by factors such as:
  - Inputs from the ML Intent to the MLFO which may impose constraints on the placement of ML pipeline nodes.

NOTE 5 – The requirement to place an ML model (M) node on a network computing resource which provides a specific type of acceleration capability is an example of constraints on the placement of an M node.

- Feedback received by the MLFO from the management and orchestration functions of the ML underlay networks or from the ML pipeline nodes may provide inputs on the placement and chaining of ML pipeline nodes.

NOTE 6 – Decoupling of the location of the ML pipeline nodes from their functionalities, except in the case of performance constraints, is achieved using the placement and chaining mechanisms.

- The deployment of an ML pipeline in future networks including IMT-2020 may span different levels. In this case, reference points between nodes located in different levels of an ML pipeline are used to allow ML pipeline multi-level distribution.

NOTE 7 – As shown in Figure 4, an ML pipeline may be distributed over multiple levels, e.g. a specific deployment may distribute the pipeline across UE, AN and CN. Based on the specific ML application, different ways of distributing the ML pipeline nodes are possible.

- **ML sandbox subsystem.** An ML sandbox subsystem allows ML pipelines to adapt to dynamic network environments such as those of future networks including IMT-2020 where a variety of conditions may change (e.g., air interface conditions, UE position, network capabilities and resources). The ML sandbox subsystem includes ML pipeline(s) and simulated ML underlay networks, and it is managed by the MLFO according to the specifications in the ML Intent. The ML sandbox subsystem allows network operators to study the effect of ML outputs before deploying them on live ML underlay networks. Feedbacks from the functions of the management subsystem are provided to the ML sandbox subsystem so that the ML pipelines of the ML sandbox subsystem can adapt to the dynamically changing network environments.

The following points are to be noted:

- The reference point between ML pipeline subsystem and ML sandbox subsystem allows the ML pipelines to interface with the ML sandbox subsystem for training and update of ML models.
- Data from the ML underlay networks and/or the simulated ML underlay networks may be used to train the ML models in the ML sandbox subsystem.
- The management of the ML pipeline nodes in the ML sandbox subsystem is also controlled by the MLFO. This allows the MLFO to train and select the ML model(s) for a given ML application.

### 8.3 General guidelines for realization of the high-level architecture

General guidelines for realization of the high-level architecture on different technology-specific underlay networks are as follows:

- Instantiation of the ML pipeline nodes: an ML application is described using ML Intent. The flow of information in an ML application can be represented by the chaining in an ML pipeline. The data from various source nodes, e.g. coming from various underlying networks, need to be gathered (by a collector node) and pre-processed (by a pre-processor node) before feeding these data to the ML model (model node). The output of the ML model is then used to apply policies (by a policy node) that will be implemented (by a SINK node).

An ML application can be realized by instantiating nodes of the ML pipeline with specific roles (e.g., SRC, C, SINK), and associating these nodes to the technology specific underlying network functions, based on the corresponding requirements of the ML application and the capabilities of the underlying network functions.

The instantiation is performed by the MLFO in coordination with the other management and orchestration functions.

- Interfacing of ML application with underlying network functions: there are two points of specific interfacing with the underlying network functions for an ML application - the SRC and the SINK nodes. The SRC nodes may have either an SBA based interface to the associated NFs which produce data or a technology-specific interface for non-SBA capable NFs. Similarly, the SINK nodes may have an SBA based interface to the associated NFs which enforce the ML output policies or a technology-specific interface for non-SBA capable NFs.
- Management of the ML pipeline: this is done by the MLFO in coordination with the other management and orchestration functions.
- ML model training and evaluation in the ML sandbox: this is controlled by the MLFO independently of the ML underlay networks. The reference point between ML sandbox subsystem and ML pipeline subsystem is used to transfer trained ML models, data for training and ML model updates between the ML sandbox subsystem and the ML underlay networks which interact with the live ML pipeline.

Appendix I gives examples of applying the above guidelines to the realization of the high-level architecture on technology-specific underlay networks.

### 9. Security considerations

This Recommendation describes the architectural framework of machine learning which is expected to be applied in future networks including IMT-2020: therefore, general network security requirements and mechanisms in IP-based networks should be applied [ITU-T Y.2701] [ITU-T Y.3101].

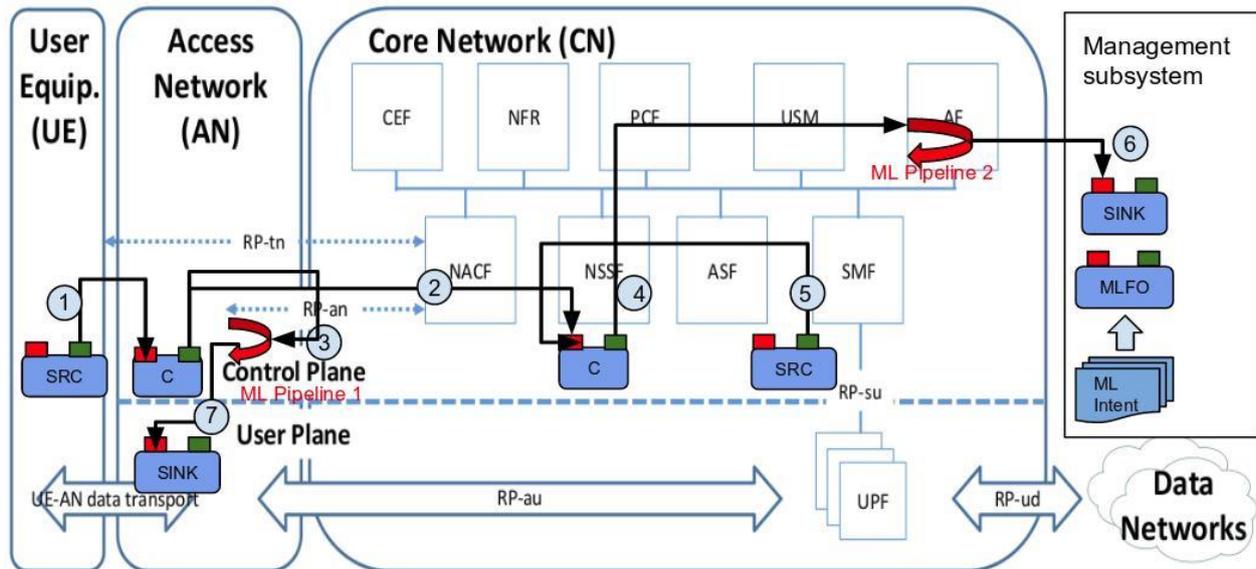
It is required to prevent from unauthorized access to, and data leaking from, an ML pipeline, whether or not they have a malicious intention, with implementation of mechanisms regarding authentication and authorization, external attack protection, etc.

## Appendix I

### Realisation of the high-level architecture on technology-specific underlay networks

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

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



**Figure I-1 – Example of realisation of the high-level architecture in an IMT- 2020 network**

This example of realisation is represented in the following manner: the ML pipeline shows the ML pipeline node positions wherever the nodes are hosted, e.g., CN, AN, UE or management functions. For example, the ML pipeline represented by arrows 1→2→4→ML pipeline 2 uses inputs from UE to make predictions at CN (e.g., MPP-based ML applications).

NOTE - The sandbox subsystem is not shown for simplicity in Figure I-1, but its functionality is applicable also to Figure I-1.

The following describes different realization instances with respect to requirements identified in clause 7:

- Realisation instance in support of requirements in clauses 7.1 and 7.2:
  - Consider arrows 5→4→ML pipeline 2→6: this ML pipeline uses inputs from CN and possibly a combination of UE inputs to make predictions at CN and applies them to the management functions. This application of ML output can in turn affect configurations in different levels (e.g., SON decisions made at CN or closed loop decisions on resource allocations done in the network).
  - Consider arrows 1→3→ML pipeline 1→7: this ML pipeline uses inputs from UE and hosts an ML model in AN for latency sensitive decisions to be applied in AN itself.
- Realisation instance in support of requirements in clause 7.3:

- Consider arrow 1: this can be realised using RRC.
- Consider arrows 2, 3, 4, 5, 7: this can be realised as an extension of reference points in CN [b-ITU-T Y.3104].
- Consider arrow 6: this can be realized via reuse of reference points defined in [ITU-T Y.3111].
- Realisation instance in support of requirements in clause 7.4:
  - UE is a resource-constrained device, hence only the SRC is instantiated in UE. This constraint is specified in the ML Intent.
- Realization instance in support of requirements in clause 7.5:
  - The collectors in AN and CN are placed by the MLFO based on the specifications of the ML applications in the ML Intent. For latency sensitive applications in AN, ML pipeline 1 is used. ML pipeline 2 is used for latency tolerant applications. The chaining is done according to the requirements specified in the ML Intent.

## Appendix II

### Mapping of the architectural components and supporting aspects to the requirements in clause 8

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

Table II-1 provides the mapping of the architectural components and supporting aspects to the requirements in clause 7.

**Table II-1 – Mapping of the architectural components and supporting aspects to the requirements**

Architectural components and supporting aspects	Requirements	Mapping explanation
ML pipeline	See clauses 7.1 and 7.2.	<p>The ML pipeline provides a common vocabulary for ML in future networks including IMT-2020. By defining the ML pipeline nodes, it becomes possible to consider the evolution of ML separately from the underlay networks.</p> <p>Instantiation of these nodes form an important part of deployment of ML overlay on different underlay networks. The functions of these nodes are defined independently of their location in the network, and hence this allows flexible placement of such nodes in the network. Split, merge and chaining of ML pipeline nodes allows deploying complex functions from these basic nodes.</p>
MLFO	See clauses 7.2, 7.4 and 7.5	<p>The MLFO orchestrates and manages the ML pipeline nodes. It is also responsible for optimal placement and chaining of ML pipeline nodes in the network. It implements these functions in coordination with the management and orchestration functions. The declarative specifications of the ML applications are supported by the MLFO by converting them into underlay-specific deployments. It also selects the models and reselects them based on the needs of the ML applications and other constraints defined in the ML Intent.</p>
ML sandbox	See clause 7.5	<p>This subsystem provides the ability to train, evaluate and monitor the performance of ML models before deploying them in a real network. It interfaces with the underlay</p>

		networks for transfer of data or trained models. In addition, it uses the simulated ML underlay networks which may generate data required for training the ML models.
ML Intent	See clause 7.4	Written in a metalanguage, this architectural supporting aspect defines the ML application, its needs in terms of data input, configuration output, any other constraints or characteristics. It is used as input towards the functionalities of the MLFO. This is the input for MLFO to create a deployable ML pipeline (one which conforms to the requirements) on specific underlay networks. By standardising this architectural aspect, it is possible for third party solution providers to integrate with the ML pipeline.
SBA	See clause 7.3	This architectural supporting aspect is used to interface between the underlay network functions and the ML pipeline. It provides the necessary loose coupling between the ML overlay and the ML underlay networks. SBA is also used to interface between the simulated ML underlay networks and the ML pipeline. It is used to train the ML pipeline and to configure the simulated ML underlay networks using the ML pipeline.

## Bibliography

- [b-ETSI MEC 003] ETSI GS MEC 003 V1.1.1 (2016), *Mobile Edge Computing (MEC); Framework and Reference 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-ETSI NFV-IFA 014] ETSI GS NFV-IFA 014 V2.3.1 (2017), *Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification*.
- [b-ETSI TS 129 500] ETSI TS 129 500 V15.0.0 (2018-07) 5G; 5G System; *Technical Realization of Service Based Architecture*.
- [b-GSMA IR.92] GSMA IR.92 (VoLTE) – *IMS Profile for Voice and SMS*.
- [b-IEC] IEC whitepaper on Edge Intelligence, [http://www.iec.ch/whitepaper/pdf/IEC\\_WP\\_Edge\\_Intelligence.pdf](http://www.iec.ch/whitepaper/pdf/IEC_WP_Edge_Intelligence.pdf).
- [b-ITU GLOSSARY] Broadband: Acronyms, Abbreviations & Industry Terms, <https://www.itu.int/osg/spu/ni/broadband/glossary.html>
- [b-ITU-T Q.5001] Recommendation ITU-T Q.5001 (2018), *Signalling requirements and architecture of intelligent edge computing*.
- [b-ITU-T Y.2322] Recommendation ITU-T Y.2322 (2018), “*The functional architecture of virtualized control network entities management and orchestrator in next generation network evolution*.”
- [b-ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), “*Security requirements for NGN release 1*”
- [b-ITU-T Y.3100] Recommendation ITU-T Y.3100 (2017), “*Terms and definitions for IMT-2020 network*”
- [b-ITU-T Y.3101] Recommendation ITU-T Y.3101 (2018), “*Requirements of the IMT-2020 network*”
- [b-ITU-T Y.3102] Recommendation ITU-T Y.3102 (2018) “*Framework of the IMT-2020 network*.”
- [b-ITU-T Y.3104] Recommendation ITU-T Y.3104 (2018), “*Architecture of the IMT-2020 network*”
- [b-ITU-T Y.3105] Recommendation ITU-T Y.3105 (2018), “*Requirements of capability exposure in the IMT-2020 network*.”
- [b-ITU-T Y.3110] Recommendation ITU-T Y.3110 (2017), “*IMT-2020 network management and orchestration requirements*”
- [b-ITU-T Y.3170] Recommendation ITU-T Y.3170 (2018), “*Requirements for machine learning-based quality of service assurance for the IMT-2020 network*.”
- [b-ITU-T Y.3650] Recommendation ITU-T Y.3650 (2018), “*Framework of big-data-driven networking*.”
- [b-3GPP TS 23.501] 3GPP TS 23501 Release 15 (2018), “*System Architecture for the 5G System*.”

[b-3GPP TS 28.554]

3GPP TS 28.554, “*Management and orchestration of 5G networks; 5G End to end Key Performance Indicators (KPI) (Release 15)*”

---