INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2022-2024

SG13-TD74/WP1

STUDY GROUP 13

Original: English

Question(s): 20/13 Geneva, 4-15 July 2022

TD

Source: Editors

Title: Draft new Recommendation ITU-T Y.ML-IMT2020-SANDBOX: “Architectural framework for Machine Learning Sandbox in future networks including IMT-2020”

Contact:	Abhay Shanker Verma Telecom Engineering Centre (TEC) India	Tel: + 91 9999554900 E-mail: as.verma@gov.in
Contact:	Vijay Kumar Roy Telecom Engineering Centre (TEC) India	Tel: +91 7011000101 E-mail: vk.roy@gov.in
Contact:	Ranjana Sivaram Telecom Engineering Centre (TEC) India	Tel: +91 9868136990 E-mail: ranjana.sivaram@gov.in
Contact:	Vishnu Ram O.V. Independent Consultant India	Tel: +91 9844178052 E-mail: vishnu.n@ieee.org
Contact:	Francesc Wilhelmi Telecommunications Technological Center of Catalunya (CTTC) Spain	Tel: +34 93 645 29 00 E-mail: fwilhelmi@cttc.cat

Keywords: High-level architecture, IMT-2020, machine learning, requirement, sandbox, simulator

Abstract: This document contains the draft new Recommendation Y.ML-IMT2020-SANDBOX “Architectural framework for Machine Learning Sandbox in future networks including IMT-2020”, output of Q20/13 meeting, Geneva, 4-15 July 2022 – for consent

This document is based on SG13- TD26/WP1 and according to the 4-15 July 2022 Q20/13 meeting's discussion and results on the following contributions:

C-#	Source	Title	Qs	Results/Notes
C161	Ministry of Communications (India)	Draft new Recommendation Y.ML-IMT2020-SANDBOX "Architectural framework for Machine Learning Sandbox in future networks	Q20/13	Accepted with modifications (4-15 July 2022) (in line with agreed modifications for C200).

		including IMT-2020"		
C200	Telecommunications Technological Center of Catalunya (Spain), India	Modifications to Draft new Recommendation Y.ML-IMT2020-SANDBOX "Architectural framework for Machine Learning Sandbox in future networks including IMT-2020"	Q20/13	Discussed and accepted with modifications (4-15 July 2022): 1. Modified the conventions to include colour legend for architecture figure and added conventions for ML pipeline as per Y.3172. 2. Simplified and clarified portions of the Introduction. 3. Modified the "NOTE"s in requirements where it can help readability and explanation. 4. Fixed editor’s notes and added Ref point 11 in the architecture figure. 5. Reordered clauses 8.3 and 8.4 on APIs and sequence diagrams for better readability. 6. Clarifications in the text to better align components, requirements and APIs.

Annexure-I

Draft new Recommendation ITU-T Y.ML-IMT2020-SANDBOX

Architectural framework for ML sandbox in future networks including IMT-2020

Summary

This Recommendation provides an architectural framework for machine learning (ML) sandbox in future networks including IMT-2020. More precisely, it describes requirements and high-level architecture for ML sandbox in future networks including IMT-2020.

Keywords

High-level architecture, IMT-2020, machine learning, requirement, sandbox, simulator

Table of Contents

Draft new Recommendation ITU-T Y.ML-IMT2020-SANDBOX

Architectural framework for ML sandbox in future networks including IMT-2020

1. Scope

This Recommendation provides an architectural framework for the ML sandbox in the context of integrating machine learning in future networks including IMT-2020. This Recommendation provides requirements and high-level architecture of ML sandbox. Architectural components along with corresponding reference points and APIs are specified.

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.3172]	ITU-T Recommendation Y.3172 (2019), "*Architectural framework for machine learning in future networks including IMT-2020*"
[ITU-T Y.3173]	ITU-T Recommendation Y.3173 (2020) *"Framework for evaluating intelligence levels of future networks including IMT-2020"*
[ITU-T Y.3174]	ITU-T Recommendation Y.3174 (2020) *"Framework for data handling to enable machine learning in future networks including IMT-2020"*
[ITU-T Y.3176]	ITU-T Recommendation Y.3176 (2020) *"Machine learning marketplace integration in future networks including IMT-2020"*
[ITU-T Y.3179]	ITU-T Recommendation Y.3179 (2021) *"Architectural framework for machine learning model serving 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 model [ITU-T Y.3172]: model created by applying machine learning techniques to data to learn from.

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

3.1.3 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.

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

3.1.5 machine learning marketplace [ITU-T Y.3176]: a component which provides capabilities facilitating the exchange and delivery of machine learning models among multiple parties.

NOTE 1 – Examples of parties include suppliers and users of ML models. Capabilities provided to users of ML models include functionalities to find, learn about, deploy (or download), and use ML models. Capabilities provided to suppliers of ML models (e.g., data scientist) include functionalities to share (on-board, upload), describe (learn about), and market their ML models.

NOTE 2 – A network operator may use a machine learning marketplace deployed internally and/or externally to the network operator's administrative domains. Internal and external marketplaces differ only in the deployment perspective. A marketplace which is internal to a network operator may act as an external marketplace to another network operator and vice versa.

3.1.6 machine learning model metadata [ITU-T Y.3176]: information which describes the characteristics of a machine learning model.

NOTE – Machine learning model metadata includes, but is not limited to, name of the ML model, ML model's author, version of the ML model, license information of the ML model, description of the data inputs and outputs of the ML model, and runtime environment of the ML model.

3.1.7 network intelligence level [ITU-T Y.3173]: level of application of automation capabilities including those enabled by the integration of artificial intelligence techniques in the network.

3.1.8 machine learning model serving [ITU-T Y.3179]: a process of preparing and deploying machine learning models in different deployment environments to enable the application of model inference to machine learning underlay networks.

3.2 Terms defined in this Recommendation

3.2.1 evaluation ML pipeline: chaining of pipeline nodes and simulated network functions (NFs) with served ML models whose goal is to evaluate a particular ML use case.

3.2.2 simulation component metadata: data describing the characteristics of a particular simulation component.

NOTE – Examples of simulation component metadata are capabilities of simulated NFs, configurable parameters, performance indicators, monitored parameters and interfaces.

3.2.3 simulation environment metadata: data describing the characteristics of a particular simulation environment.

NOTE 1 – Simulation environment metadata can contain information such as installation/execution requirements, simulation component metadata, performance indicators, connections, and maturity indicators (e.g., alpha/beta versions).

NOTE 2 – Examples of format for representing simulation environment metadata are JavaScript object notation (JSON) [b-JSON], comma-separated values (CSV) [b-CSV], or extensible markup language (XML) [b-XML].

3.2.4 simulation profile: a list of parameters and their values which describe the ML use case to be trained, evaluated, or tested at the ML sandbox.

NOTE – The list of parameters and their values may be derived from ML intent [ITU-T Y.3172] and simulation environment metadata.

4. Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AF	Application Function
AI	Artificial Intelligence

AP	Access Point
API	Application Programming Interface
DBr	Data Broker
DH	Data Handling
DM	Data Model
GAN	Generative Adversarial Network
KPI	Key Performance Indicator
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
NF	Network Function
OAM	Operation Administration and Maintenance
RAN	Radio Access Network
RL	Reinforcement Learning
SL	Supervised Learning
UE	User Equipment
UL	Unsupervised Learning
uRLLC	Ultra-Reliable Low-Latency Communication
V2X	Vehicle-to-Everything
WLAN	Wireless Local Area Network

5. Conventions

In this Recommendation:

- The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.
- The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.
- The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option, and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.
- The color “solid blue” is used in Figure 2 and Figure 3 to indicate components and interfaces that are newly defined in this document.
- ML pipeline – In this Recommendation, in alignment with the conventions of [ITU-T Y.3172] 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

6. Introduction

The integration of artificial intelligence (AI) and machine learning (ML) has been identified as one of the key features of future networks. However, network operators have the challenge of maintaining the operational performance and associated key performance indicators during or after this integration. In addition, the introduction of ML techniques to IMT-2020 networks may raise concerns regarding transparency, reliability, and availability of ML methods, techniques, and data.

Studying the trade-offs, advantages, and disadvantages while integrating various ML mechanisms is important to understand their impact on the network. For example, reducing the generalization error is the main concern in applying any kind of supervised learning (SL) approach, which can be high even if the test error is kept low (this phenomenon is commonly known as *overfitting*). Similarly, unsupervised learning (UL) aims to find patterns from data without any guidance (unlabelled data) and hence lacks validation. On the other hand, reinforcement learning (RL) is based on the learning-by-experience paradigm. RL has been shown to be of great utility for single-agent approaches in controlled scenarios, however notable adverse effects can appear as a result of the competition raised by multiple systems sharing the same resources (e.g., while providing heterogeneous services using common network resources).

Thus, all kinds of learning can lead to unexpected and/or undesired behaviour in live networks. Even if the performance of networking systems can be improved by ML techniques in the long term, it is safe to assume that the system will unavoidably experience certain performance degradation during a transitory regime. In some situations, this degradation of key performance indicators (KPI) may be unacceptable for network operators, especially for demanding requirements of certain network-oriented applications such as ultra-reliable low-latency communication (uRLLC) applications. In other cases, the network may change quickly and may not reach a stable, long-term regime that is expected to optimize the network’s performance.

NOTE – The transitory regime precedes the stability phase of an ML model when applied to a network. Performance degradation can result from potential delays in serving models in the network, or from trying suboptimal configurations during exploration periods in online learning.

Given the instability that ML methods can generate in communications systems, which can be particularly exacerbated in online mechanisms including exploration phases, the sandbox subsystem [ITU-T Y.3172] emerges as a promising solution for training, testing, and evaluating the performance of ML models before being deployed in live networks. The ML sandbox is an isolated environment in which machine learning models can be evaluated. The ML sandbox is therefore meant to reproduce the behaviour/operation of live networking systems, thus improving the robustness and resilience of future ML-enabled networking systems. ML sandbox includes a managed test network (e.g., a testbed) or a software-based environment (e.g., using a simulator or emulator). Software-based network environments can be particularly useful to overcome the limitations of limited training data sets and laboratory-based testbeds. For instance, simulators can be used to frame cases that have not been noticed before (i.e., anomalies), which would contribute to enabling failure prediction, anomaly detection, and self-healing.

Through the management subsystem, network operators can manage the ML sandbox and thereby address the challenges posed by ML-driven solutions for networks. The interfaces between the machine learning function orchestrator (MLFO) and the ML sandbox allow the manageability of the replicated network environment (e.g., simulation), the execution of test cases, and the evaluation of ML models.

7. Requirements

The requirements for the ML sandbox's architectural framework are divided into the following categories:

- Simulated ML underlay requirements
- Operational requirements
- Communication requirements
- Metadata requirements

7.1 Simulated ML underlay requirements

REQ-ML-SANDBOX-001: The ML sandbox is required to simulate heterogeneous sources of data (SRCs) and sinks (SINKs) of ML output.

NOTE 1 – SRCs and SINKs simulated in the ML sandbox include those within the IMT-2020 network as well as application functionalities hosted in network slices. Examples of application functionalities hosted in network slices are vehicle-to-everything (V2X) applications, Industry 4.0 applications, and emergency applications.

REQ-ML-SANDBOX-002: The ML sandbox is required to support the dynamic instantiation of new simulated SRCs and/or SINK nodes.

NOTE 2 – Instantiation of new simulated SRCs and SINK nodes is managed by MLFO.

REQ-ML-SANDBOX-003: The ML sandbox is required to consider policy inputs from the operator while configuring the simulated ML underlay networks [ITU-T Y.3172].

NOTE 3 – Examples of policy inputs are those related to conflict resolution and resource management.

REQ-ML-SANDBOX-004: The ML sandbox is required to enable coordinated time synchronization of operations executed in the ML sandbox as required by the specific use case.

NOTE 4 – The time synchronization may be coordinated by the MLFO by controlling the sequence of operations executed in the ML sandbox. The sequence of operations triggered by the MLFO may be according to the synchronisation requirements of the specific use case. An example of a sequence of operations triggered by the MLFO is, generation of data by radio access network (RAN)-specific simulator which is input into the corresponding ML model as SRC, followed by analysis in the ML model, and finally application of ML inference into specific simulators for SINK.

REQ-ML-SANDBOX-005: The ML sandbox is recommended to consider the quality of data needed for ML models (training or testing) while generating the simulated data.

NOTE 5 – The quality of data depends on the use case requirements. The requirements on the quality are input in the ML intent. Examples are alignment and similarity with live networks, including user equipment (UE) capabilities, granularity of reported UE measurements, frequency of channel measurements, accuracy of measured parameters, etc.

REQ-ML-SANDBOX-006: The ML sandbox is recommended to support demand mapping [ITU-T Y.3173] for configuring and updating the simulated ML underlay networks.

NOTE 6 – Demand mapping is achieved by continuous, run-time, matching of the ML intent with the configuration options provided by the simulated ML underlay network. The configuration of the simulated ML underlay networks may be continuously adjusted based on demand mapping.

NOTE 7 – Demand mapping may be implemented through the analysis of data patterns and ML pipeline output and corresponding optimization of simulated ML underlay networks.

REQ-ML-SANDBOX-007: The ML sandbox is required to provide sanity checks to assess the correct operation of the simulated ML underlay networks.

7.2 Operational requirements

REQ-ML-SANDBOX-008: The ML sandbox is required to support multiple evaluation ML pipelines, which may be chained and interfaced with simulators from different levels of the network.

NOTE 1 – Network levels are defined in [ITU-T Y.3172].

REQ-ML-SANDBOX-009: The ML sandbox is required to support the monitoring and evaluation of ML pipelines and simulation components according to specifications in the ML intent.

NOTE 2 – Examples of monitoring and evaluation output may include threshold-based asynchronous notifications from the ML sandbox (to the MLFO), post-processing of ML output, metering, security threat analysis, etc. Other output may include updated metadata which reflects the results of the evaluations of the models in the ML sandbox.

REQ-ML-SANDBOX-010: The ML sandbox is required to support the testing and evaluation of multiple ML pipelines at the same time, including aggregated impacts on the network due to them.

NOTE 3 – For example, different ML pipelines may use different types of models (e.g., based on RL and SL). The type of model may be decided by the MLFO based on the use case. Simultaneous evaluation of the different ML pipelines may be triggered for addressing an ML use case. The outputs of these ML pipelines may be compared to make an optimal decision.

REQ-ML-SANDBOX-011: The ML sandbox is required to support training and testing ML models that combine simulated and real data from the network.

NOTE 4 – The choice of data to be used is managed by the MLFO [ITU-T Y.3172].

NOTE 5 – The combination of simulated and real data may also include augmented data.

REQ-ML-SANDBOX-012: The ML sandbox is required to support dynamic resource management for ML pipeline nodes instantiated in the ML sandbox.

NOTE 6 –The instances of ML pipeline nodes in the ML sandbox (e.g., simulated SRC node) may need resource management mechanisms like dynamic resource allocation. The ML sandbox may use various request handling mechanisms like load balancing towards ML pipeline nodes (e.g., ML model) in the ML sandbox.

REQ-ML-SANDBOX-013: The ML sandbox is required to enable granular evaluation of ML test cases.

NOTE 7 – In the case of batch jobs (combined test cases) which are triggered by the ML sandbox, isolation of problems found in the evaluation stage need granular information on the specific test case which failed. The ML sandbox is needed to enable such granular evaluation.

REQ-ML-SANDBOX-014: The ML sandbox is required to support monitoring and evaluating the network intelligence level.

NOTE 8 – See [ITU-T Y.3173] for monitoring and evaluating network intelligence level.

REQ-ML-SANDBOX-015: The ML sandbox is required to support testing techniques to enhance the robustness of the ML pipelines.

NOTE 9 – Examples of testing techniques include regression and/or integration testing techniques for testing ML models, data generation techniques for ensuring quality and augmentation of simulated data, simulation of failure scenarios, or rare scenarios for ML model training.

REQ-ML-SANDBOX-016: The ML sandbox is required to produce the output of simulations, tests, and evaluations in a machine-readable format.

NOTE 10 – Metadata corresponding to the model may be updated with the results of the evaluations. Such updated metadata may be used by MLFO in future selections of models.

7.3 Communication requirements

REQ-ML-SANDBOX-017: The ML sandbox is required to support data handling (DH) reference points toward technology-specific simulated ML underlay networks.

NOTE 1 – Data handling reference points are defined in [ITU-T Y.3174].

REQ-ML-SANDBOX-018: The ML sandbox is required to support the transfer of trained models across the different ML pipelines in the sandbox as well as to other subsystems in the ML overlay.

NOTE 2 – Application and reuse of trained models from the ML sandbox for many use cases are examples of scenarios where the transfer of trained models across different ML pipelines in the ML sandbox is required. The transfer and deployment of trained models in live networks to enable specific use cases is an example of a scenario that requires the transfer of trained models from the ML sandbox to other ML overlays.

REQ-ML-SANDBOX-019: The ML sandbox is required to support the transfer of data for training or testing models across different ML pipelines in the sandbox as well as to other ML overlays.

REQ-ML-SANDBOX-020: The ML sandbox is required to support interfaces with ML marketplaces to transfer ML models and corresponding metadata.

NOTE 3 – See reference point 13 in [ITU-T Y.3176] for the interface between ML marketplaces and the ML sandbox. This interface serves both in the downlink (e.g., download models) and the uplink (e.g., update models).

NOTE 4 – An example of metadata is the outcome of applying an ML model in a live or test network, which can be used to enhance trust and confidence in an ML model available in the marketplace.

REQ-ML-SANDBOX-021: The ML sandbox is required to support data handling mechanisms including metadata storage, communication interfaces with data models and ML underlay networks, and data storage.

NOTE 5 – See [ITU-T Y.3174] for data handling mechanisms.

7.4 Metadata requirements

REQ-ML-SANDBOX-022: The ML sandbox is recommended to reuse the ML metadata store across different ML underlay networks to allow interworking between evaluation ML pipelines and simulated ML underlay networks.

NOTE 1 – API-g is stored in the management subsystem to allow the training, testing, and evaluation of ML models in the simulated ML pipeline [ITU-T Y.3174].

NOTE 2 – DM and corresponding API-s used in the simulated ML underlay network are stored in the management subsystem to allow the interworking between the data broker (DBr) and the simulated NFs [ITU-T Y.3174].

REQ-ML-SANDBOX-023: The ML sandbox is recommended to derive the simulation profile from ML intent inputs from the MLFO along with the simulation environment metadata and use it to configure and update the simulated ML underlay networks.

NOTE 3 – The simulation profile may include a list of parameters and their values which describe the ML use case to be trained, evaluated, or tested at the ML sandbox. The MLFO can provide ML intent inputs offline or at runtime, based on triggers.

NOTE 4 – The simulation environment metadata describes the parameters of each simulator. This is provided by the simulation designer.

REQ-ML-SANDBOX-024: The ML sandbox is recommended to use ML model metadata from the ML Marketplace to adjust the simulated ML underlay networks and the evaluation scenarios.

NOTE 5 – For instance, the limitations of algorithms in terms of the amount of data (e.g., unsupervised learning) should be input as the amount of data to be generated (e.g., the number of access points (AP) to be simulated, the total simulation time, the minimum number of events, etc.).

REQ-ML-SANDBOX-025: The ML sandbox is required to support simulation environment metadata.

NOTE 6 – Simulation environment metadata can be provided to the serving framework for considering the deployment environment while creating an inference engine (see clause 8.1.3 in [ITU-T Y.3179]).

NOTE 7 – Simulation environment metadata includes the data models used by simulated NFs and APIs to access these data.

NOTE 8 – Simulation environment metadata can be used by the data handling to select the type of storage of data (REQ-ML-DH-011 in [ITU-T Y.3174]).

REQ-ML-SANDBOX-026: The ML sandbox is recommended to support isolation between different instances of evaluation ML pipelines (instantiated for different ML underlay networks).

NOTE 9 – Examples of reasons for isolation are security, data privacy reasons, and support for slicing.

8. High-level architecture

The high-level architecture of the ML sandbox is described here in the context of architecture frameworks described in [ITU-T Y.3172], [ITU-T Y.3174] and [ITU-T Y.3179]. Interactions between the components of the ML sandbox subsystem and other components of the architecture framework are elaborated with a specific focus on modifications to reference points. The components of the ML sandbox subsystem and their functionalities are described.

8.1 ML sandbox within the high-level ML architecture

To simulate ML underlay networks, the ML sandbox includes simulated NFs, application functions (AFs), and ML pipeline(s) whose elements are managed by the MLFO [ITU-T Y.3172]. The ML sandbox is particularly useful to address dynamic networking systems since it allows validating the effect of ML-based optimizations before being deployed in production environments. Besides, because of the potential limitations of data coming from live networks (insufficient amount, privacy issues, etc.), the ML sandbox can be used to generate synthetic data as a complement for a given training procedure.

Figure provides the high-level architecture showing the main involved components and the ML sandbox, which are intended to fulfil the requirements specified in clause 7.

NOTE 1– See clause 8.2 for further details regarding the ML sandbox architectural components shown in Figure 2.

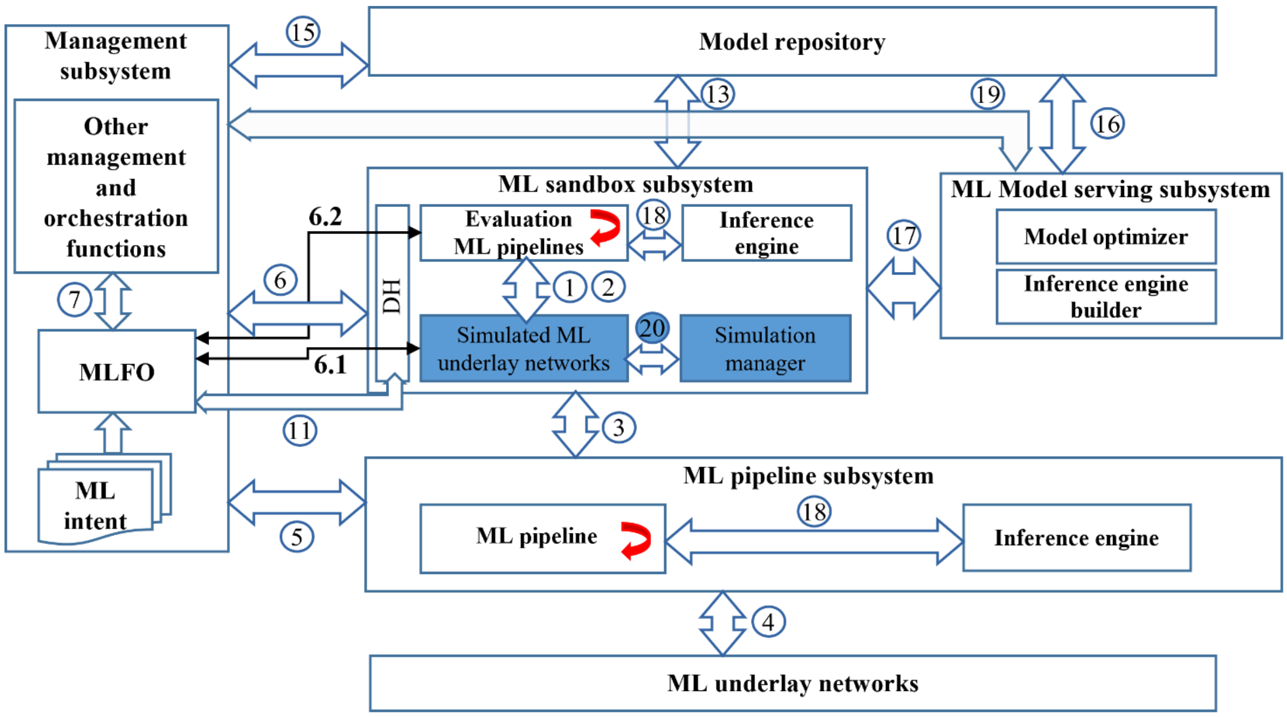

Figure 2: ML sandbox within the high-level ML architecture

Figure 2 showcases the ML sandbox subsystem and its main components in the context of the high-level architectural framework defined in [ITU-T Y.3172]. It extends the high-level architecture for ML model serving [ITU-T Y.3179] with specific architecture components of the ML sandbox and their corresponding interactions.

The reference points shown in Figure 2 are as follows:

Reference points 1 and 2 act as internal reference points within the ML sandbox subsystem, between the simulated ML underlay networks and the evaluation ML pipeline, and are used unmodified, as defined in [ITU-T Y.3172], for training and update of ML models at the ML sandbox subsystem.

Reference point 3 is the reference point between the ML sandbox and ML pipeline subsystems [ITU-T Y.3172]. It allows the ML pipelines to interface with the ML sandbox subsystem for training and update of ML models. It is used only as a model management interface, as described in [ITU-T Y.3179].

Reference point 4, as defined in [ITU-T Y.3174], is the interface between the ML pipeline and the ML underlay network. It is used for the transfer of data between the ML underlay network and the (evaluation) ML pipeline instantiated in the ML sandbox (see clause 8.2 in [ITU-T Y.3174]). 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.

Reference point 5, as defined in [ITU-T Y.3172], is the interface between the management subsystem and the ML pipeline subsystem.

Reference point 6 is used for the management subsystem to manage the models applied to the ML sandbox [[ITU-T Y.3172], including monitoring and evaluating network intelligence levels [ITU-T Y.3173]. Reference point 6 has two parts:

- Reference point 6.1 [ITU-T Y.3174] is the interface between the management subsystem and the simulated ML underlay network to orchestrate and manage simulated ML underlay networks.

- Reference point 6.2 interfaces the management subsystem with the evaluation ML pipeline to orchestrate and manage the evaluation ML pipeline.

  NOTE 2– Data from the ML underlay networks and/or the simulated ML underlay networks may be used to train ML models in the ML sandbox subsystem.

Reference point 7 is the interface between MLFO and other management and orchestration functions of the management subsystem, used unmodified as defined in [ITU-T Y.3172].

Reference point 11 is the interface between the MLFO and the data handling components in the ML overlay, used unmodified as defined in [ITU-T Y.3174].

Reference point 13 is the interface between the ML marketplace and the ML sandbox subsystem, used unmodified as defined in [ITU-T Y.3176].

Reference point 15 is the interface between the management subsystem and the ML marketplace, used unmodified as defined in [ITU-T Y.3176].

Reference point 16 is the interface between ML model serving subsystem and model repository, used unmodified as defined in [ITU-T Y.3179].

Reference point 17 is the interface between the ML sandbox subsystem and ML model serving subsystem, used unmodified as defined in [ITU-T Y.3179].

Reference point 18 is the interface between evaluation ML pipelines and the inference engine, used from [ITU-T Y.3179].

NOTE 3 – The evaluation ML pipeline referred to here is the same as the ML pipeline in the ML sandbox subsystem in [ITU-T Y.3179].

Reference point 19 is the interface between the management subsystem and the ML model serving subsystem, used unmodified as defined in [ITU-T Y.3179].

Reference point 20 is the interface between the simulated ML underlay networks and the simulation manager used for managing the simulated ML underlay networks. See clause 7.1 for more details.

8.2 Components of the ML sandbox

The ML sandbox contains the components defined in the following sub-clauses. The detailed architecture of the ML sandbox subsystem is illustrated in Figure 3.

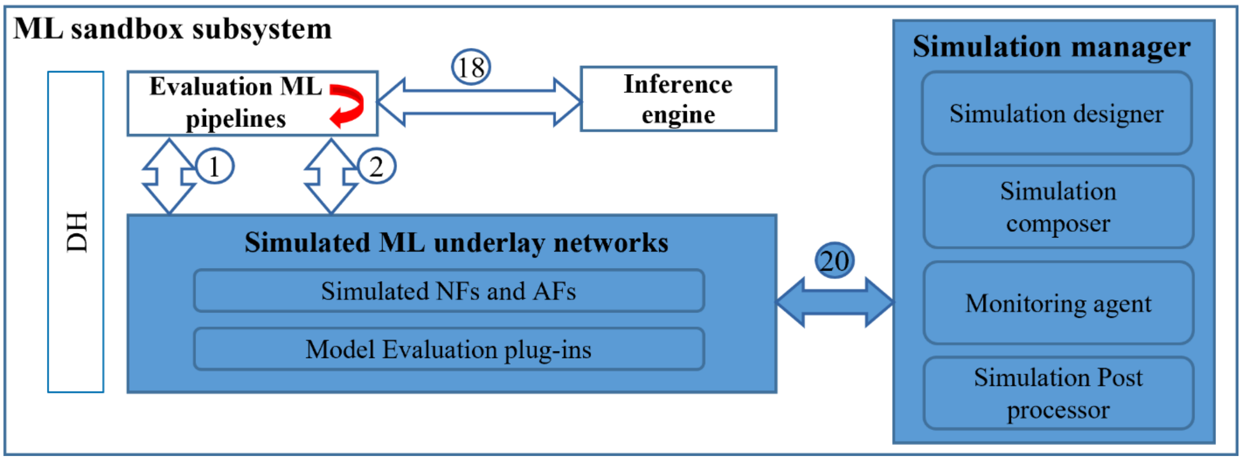

Figure 3: Detailed architecture of the ML sandbox subsystem

8.2.1 Simulated ML underlay networks

The simulated ML underlay networks component is reused from [ITU-T Y.3172]. As explained in [ITU-T Y.3172], the ML sandbox can use data generated from simulated ML underlay networks (obtained via reference points 1 and 2), and/or live networks (obtained via reference point 3), for training or testing of ML models.

In this document, two subcomponents of simulated ML underlay networks are introduced, simulated NFs and AFs, and model evaluation plug-ins.

NOTE – An example of a simulated NF is a third-party simulation tool such as ns-3 [b-Riley-ns3].

8.2.1.1 Simulated NFs and AFs

As explained in [ITU-T Y.3174], simulated NFs and AFs provide the ability to support heterogenous sources (SRC) of data and SINK functionality. In this document, these SRC and SINK are used for training and testing the evaluation ML pipelines.

8.2.1.2 Model evaluation plug-ins

Model evaluation plug-ins are responsible for evaluating the performance of ML models as per the requirements defined in the use case. The plug-ins interact with the simulated NFs and AFs using technology-agnostic interfaces, which would enable the interaction with heterogeneous third-party applications such as network simulators and the collection of ML model evaluation parameters.

NOTE – Examples of ML model KPIs are model accuracy, recall, and precision. Other parameters evaluated could be inference latency and memory footprint. These parameters are to be specified in the use case description provided in the ML intent [ITU-T Y.3172].

8.2.2 Simulation manager

The simulation manager manages the simulated ML underlay networks, specifically consisting of the following subcomponents: simulation designer, simulation composer, monitoring agent, and simulation post-processor.

Based on inputs from MLFO, the simulator manager takes into account metadata and policy inputs from the operator while managing the simulated ML underlay networks. The simulation manager is responsible for achieving demand mapping [ITU-T Y.3173] while configuring and updating the simulated ML underlay networks. The simulation manager provides dynamic resource management for ML pipeline nodes instantiated in the ML sandbox.

8.2.2.1 Simulation designer

Based on the input from MLFO regarding the simulation requirements for the use case, the simulation designer prepares the set of simulation resources that compose the simulated ML underlay, corresponding to the ML use case.

NOTE 1 – Inputs from MLFO may include time-synchronization of operations executed in the ML sandbox as required by the specific use case.

NOTE 2 – The information to design simulated ML underlays can come from the use case (ML intent) or data gathered by the live ML underlay.

NOTE 3 – As an example, in a traffic steering use case, ML models may be applied to predictively managing the resource allocation in the network. The simulation designer arrives at the simulation needs for this use case which may include data generation and simulated resource management mechanisms and corresponding parameters and configurations. As another example, as a result of network dynamics, some path-loss parameters used by the ML sandbox subsystem may vary over time. To address this issue, the MLFO keeps track of those changes and provides feedback to the simulation designer to update the necessary simulation parameters.

8.2.2.2 Simulation composer

The simulation composer uses the design from the simulation designer and identifies the specific simulation components to use for the use case. It takes as input the configurations and KPIs as specified in the use case. The simulation composer then deploys, installs, and instantiates the simulated ML underlay components (e.g., NFs, AFs and model evaluation plug-ins) to be used for simulating different types of network underlays and evaluating various types of ML models. The simulation composer may chain and interface simulators from different levels of the network [ITU-T Y.3172] with multiple ML pipelines.

NOTE – Based on the specified role and requirements of the simulation (derived from use cases), the simulation composer may indicate the best simulation tool (e.g., a specific implementation of a RAN simulator). Specific testing techniques like robustness testing may be applied.

8.2.2.3 Monitoring agent

The monitoring agent monitors and evaluates the simulations in the ML sandbox, including the evaluation ML pipeline and the simulated ML underlay network. The monitoring agent enables granular evaluation of ML test cases by MLFO.

In addition to the use-case-specific parameters obtained from MLFO, the following five dimensions are considered [ITU-T Y.3173]: demand mapping, data collection, analysis, decision, and action implementation.

The monitoring of data collection, action implementation, and analysis is done by the monitoring agent (see clause 8.3 of [ITU-T Y.3173]). This may include monitoring the quality of data needed for ML models (training or testing) while generating the simulated data, and sanity checks to assess the correct operation of the simulated ML underlay networks.

8.2.2.4 Simulation post-processor

The simulation post-processor provides an interface whereby data from simulated ML underlays are post-processed and presented in a standard-compliant manner to the MLFO, which performs model evaluations and/or (re)training. This step is critical to handle heterogeneous sources of information.

NOTE – For instance, once the handler gets the raw logs generated by a simulator (e.g., a CSV file), the post-processor extracts the relevant information to be used by the MLFO.

8.2.3 Evaluation ML pipeline

The evaluation ML pipeline is used for model evaluation in the ML sandbox environment as described in [ITU-T Y.3172]. This component supports the transfer of evaluated and tested models and supports interfaces with ML marketplaces to transfer ML models and corresponding metadata, in coordination with the MLFO.

NOTE – The evaluation ML pipeline is similar to the ML pipeline defined in [ITU-T Y.3172], except that it uses simulated ML underlay networks instead of live ML underlay networks. For example, both the evaluation ML pipeline and live ML pipeline use reference point 4 and reference point 5.

8.2.4 Data handling

Data handling provides functionality for storage of data models and data for simulated ML underlay networks, used unmodified as defined in [ITU-T Y.3174]. Components of DH as defined in clause 8.2 of [ITU-T Y.3174] are instantiated in the ML sandbox subsystem. Reference points 1 and 2 are reused from [ITU-T Y.3172] and [ITU-T Y.3174] between evaluation ML pipelines and simulated

ML underlay networks. The ML sandbox utilizes DH to dynamically instantiate new simulated SRCs and/or SINK nodes.

NOTE – DH is shown here for completeness. The role and interactions with DH remain the same as defined in [ITU-T Y.3174], with the only difference of addressing evaluation ML pipelines with respect to what is covered in [ITU-T Y.3174].

8.2.5 Inference engine

The inference engine provides ML model inference capability for ML pipeline(s), used unmodified as defined in [ITU-T Y.3179].

8.3 APIs

Reference points 6.1 and 6.2 were shown in Figure 2 and introduced in clause 8.1. The realization of the requirements of the ML sandbox necessitates interaction between the ML sandbox and various other components of the high-level architecture. Reference points 6.1 and 6.2 enable APIs which are used for such interactions.

The specific APIs that correspond to each reference point are described below.

NOTE – Interactions between the components using the APIs defined in this clause are depicted in the sequence diagrams in clause 8.4.

8.3.1 Reference point 6.1

8.3.1.1 Capability discovery request API (Capability_Discovery)

API description: Using reference point 6.1 and complementary external interfaces with simulation capabilities, the Capability_Discovery API discovers third-party simulation components that can be used to perform use-case-specific simulations in the ML sandbox. According to the ML use case, the MLFO finds and selects the candidate simulation environments from a list of updated capabilities.

Capability_Discovery-Request:

Direction: MLFO → ML sandbox subsystem

Table 8-1 describes the information elements of Capability_Discovery-Request.

Table 8-1 – Capability_Discovery-Request information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Request identifier	Integer	Mandatory	Identifier of the request, indicating “capability discovery”
ML profile	<Attribute, value> array	Mandatory	Includes metadata defining policies, requirements, constraints, etc.

Capability_Discovery-Response:

Direction: ML sandbox subsystem → MLFO

Table 8-2 describes the information elements of Capability_Discovery-Response.

Table 8-2 – Capability_Discovery-Response information elements

Information element	Type	Mandatory/Optional /Conditional	Description
List of simulation components	<Attribute, value> array	Mandatory	Updated capability list of the available simulation components

Simulation environment metadata	<Attribute, value> array	Mandatory	Metadata can contain information such as installation/execution requirements, capabilities of simulated NFs, performance indicators, configurable parameters, maturity (alpha/beta), etc.

NOTE 1 – As an example of third-party NF, ns-3 can be selected to simulate a specific deployment of IEEE 802.11ax Wireless Local Area Networks (WLANs) [b-IEEE 802.11].

NOTE 2 – Based on the use case requirements, the list of potential NFs is narrowed. For instance, NFs can have associated information (via simulation environment metadata) such as “running time”, “billing aspects”, “accuracy”, etc.

8.3.1.2 Status reporting API (Monitor_Reporting)

API description: Using reference point 6.1, the Status reporting API reports the status of the simulation components (e.g., health status), so that the MLFO can consider taking healing actions.

Monitor_Reporting (periodical or responsive):

Direction: ML sandbox subsystem → MLFO

Table 8-3 describes the information elements of Monitor_Reporting.

Table 8-3 – Monitor_Reporting information elements

Information element	**Type**	**Mandatory/Optional /Conditional**	**Description**
Notification identifier	Integer	Mandatory	Identifier of the request, indicating “monitoring report”
Status	Integer	Mandatory	Code indicating the current status of the simulation components, e.g., health status indicated by Green, Yellow, Orange, Red
Severity	Integer	Optional	Code indicating the severity of the potential anomalies identified (Critical, Major, Minor, Normal, or Clear)
Monitoring logs	String list	Optional	Raw data resulting from monitoring
Alerts	<Attribute, value> array	Optional	Threshold-based alerts
Suggested action points	String list	Optional	List of suggested action points to fix the potential reported issues

NOTE - Monitoring is carried out based on continuous flow of data generated by simulation components, including simulation data (SRC & SINK nodes), regression tests, reporting from simulation modules, etc.

8.3.1.3 Input/Output validation reporting API (Report_IO_Validation)

API description: Using reference point 6.1, the Report_IO_Validation API is used to report the status of input/output data used/generated at/by the ML sandbox. This information can be used by the MLFO to generate new data sets, re-train ML models with different configurations, update simulation components, etc.

Report_IO_Validation:

Direction: ML sandbox subsystem → MLFO

Table 8-4 describes the information elements of Report_IO_Validation.

Table 8-4 – Report_IO_Validation -Response information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Notification identifier	Integer	Mandatory	Identifier of the request, indicating "input/output validation result"
Validation type	Integer	Mandatory	Indicates the type of validation performed (e.g., input data to configure simulation parameters or output training data validity)
Result of validation	Integer	Mandatory	"Success" or "fail"
Warnings	String list	Optional	Detailed information regarding potential issues or misbehaviors observed from the current input/output data
Error details	String list	Conditional	Detailed information regarding the errors thrown during the validation procedure

NOTE – The data to be validated includes input data (e.g., to check that demand mapping can be fulfilled at the simulated ML underlay) and simulation output data (e.g., to assess the feasibility of trained models, the accuracy of generated data, etc.). For instance, testing techniques such as equivalence partitioning or centroid positioning [b-Zhang-Testing] can be applied to validate the diversity and the quality of the data generated by simulators (e.g., as for validating the synthetic data).

8.3.1.4 Sandbox asynchronous messages API (Sandbox_Async)

API description: Using reference point 6.1, the Sandbox_Async API is used for the sandbox asynchronous messages defined in clause 8.4.5.

Sandbox_Async:

Direction: ML sandbox subsystem → MLFO

Table 8-5 describes the information elements of Sandbox_Async.

Table 8-5 – Sandbox_Async information elements

Information element	Type	Mandatory/Optional /Conditional	Description
Message identifier	Integer	Mandatory	Identifier of the message, indicating "Sandbox asynchronous message"
Message code	Integer	Mandatory	Code of the asynchronous message type
Additional information	String list	Conditional	Additional information related to the message type

8.3.2 Reference point 6.2

8.3.2.1 MLFO-triggered operations API (MLFO_Trigger)

API description: Using reference point 6.2, the MLFO_Trigger API is used for the MLFO-triggered operations defined in clause 8.4.4.

MLFO_Trigger-Request:

Direction: MLFO → ML sandbox subsystem

Table 8-6 describes the information elements of MLFO_Trigger-Request.

Table 8-6 – MLFO_Trigger-Request information elements

Information element	Type	Mandatory/Optional /Conditional	Description

Message identifier	Integer	Mandatory	Identifier of the message, indicating “ML-triggered operation”
Operation code	Integer	Mandatory	Code of the operation to be performed
Policies & requirements	<Attribute, value> array	Conditional	Metadata including policies, requirements.
Simulation environment metadata	<Attribute, value> array	Conditional	Metadata including simulation configuration, available resources, time constraints, etc.

MLFO_Trigger-Response:

Direction: ML sandbox subsystem → MLFO

Table 8-7 describes the information elements of MLFO_Trigger-Response.

Table 8-7 – MLFO_Trigger-Response information elements

Information element	**Type**	**Mandatory/Optional /Conditional**	**Description**
Message identifier	Integer	Mandatory	Identifier of the message, indicating “ML-triggered operation”
Response code	Integer	Mandatory	Code of the operation response (OK, Bad request, Error, etc.)
Response data	(variable)	Conditional	Depending on the request type, different response data types can be provided (e.g., training data set, trained ML model, validated ML model)

8.4 Sequence Diagrams

This clause provides sequence diagrams that result from the ML sandbox operation. The sequence diagrams are derived from the requirements in clause 7, the architectural framework defined in clause 8.1 and the APIs in clause 8.3.

8.4.1 Capability discovery

Simulation components provided by third parties can be used to perform use case-specific simulations in the ML sandbox. This procedure enables the discovery of such simulation components stored in third-party repositories. The sequence diagram is shown in Figure 4.

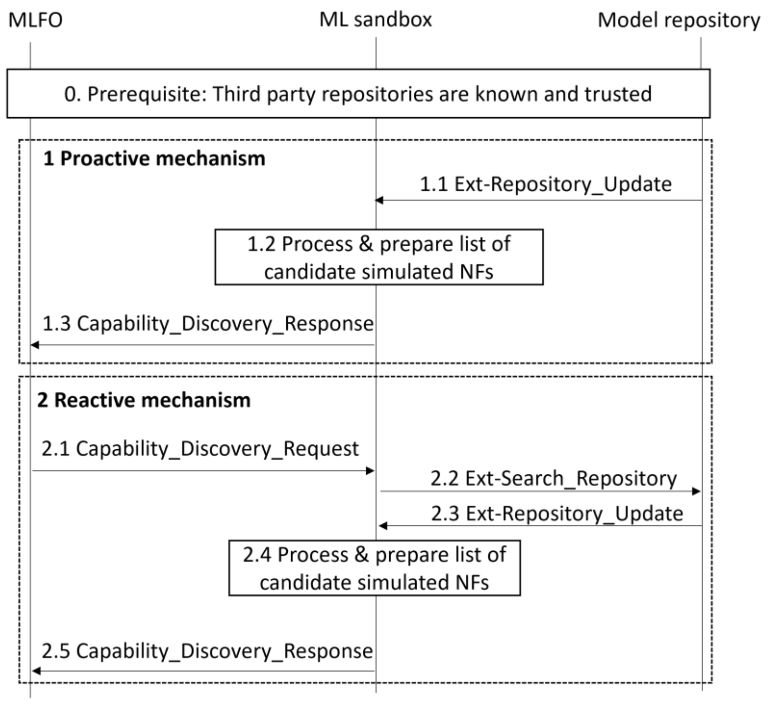
MLFO
ML sandbox
Model repository
0. Prerequisite: Third party repositories are known and trusted
1 Proactive mechanism
1.1 Ext-Repository_Update
1.2 Process & prepare list of candidate simulated NFs
1.3 Capability_Discovery_Response
2 Reactive mechanism
2.1 Capability_Discovery_Request
2.2 Ext-Search_Repository
2.3 Ext-Repository_Update
2.4 Process & prepare list of candidate simulated NFs
2.5 Capability_Discovery_Response

Figure 4: Capability discovery for third-party simulation components

Prerequisite: MLFO knows the list of third-party repositories, offline configured, trusted, secure channels. Simulation components in the repositories are described using simulation environment metadata.

Two mechanisms (*proactive* and *reactive*) are considered according to the nature of the capability discovery notification. The steps in Figure 4 are explained below.

Proactive mechanism:

This includes the following steps, as shown in Figure 4:

1.1 Third-party simulator repositories update their simulation components to the ML sandbox. These are to be evaluated in the ML sandbox in combination with the model evaluation plug-ins. Further component updates can also be provided to the ML sandbox as and when a third-party simulation repository (e.g., ns-3) releases specific features (e.g., MIMO support).

NOTE 1 – This step is done using an external interface, referred to as Ext-Repository_Update.

1.2 Information from the update message is processed to prepare an updated list of candidate simulated NFs for evaluation in the ML sandbox.

NOTE 2 – This step is done by the simulation designer (see clause 8.2.2.1).

1.3 The corresponding updated simulation components are published to trusted MLFOs.

Reactive mechanism:

This includes the following steps as shown in Figure 4:

2.1 MLFO queries the ML sandbox for simulation capabilities based on simulation environment metadata.

2.2 The ML sandbox sends the query to trusted third-party simulation component repositories.

NOTE 3 – This step is done using an external interface, referred to as Ext-Search_Repository.

2.3 Repositories respond with a list of simulation components matching the query.

NOTE 4 – This step is done using an external interface, referred to as Ext-Repository_Update.

2.4 Information from the response message is processed to prepare an updated list of candidate simulated NFs for evaluation in the ML sandbox.

NOTE 5 – This step is done by the simulation designer (see clause 8.2.2.1).

2.5 The corresponding updated simulation components are published to trusted MLFOs.

Based on the use case requirements and the result of the capability discovery mechanism, MLFO arrives at candidate simulation environments (list of NFs, simulation components, corresponding configurations, connections, data handling adaptors). From the candidate simulation environments provided by MLFO, an operator selects an optimal configuration and deploys it in the ML sandbox.

In addition, data handling and other underlay changes are also applied based on the selected configuration.

NOTE 9 – Simulation resources in the repositories are described using simulation environment metadata.

NOTE 10 – An example of third-party simulated NF is ns-3, used to simulate a specific deployment of IEEE 802.11ax WLANs.

NOTE 11 – Based on the use case requirements, the list of potential NFs is decided. NFs can have associated information (as part of simulation environment metadata) such as “running time”, “billing aspects”, “accuracy”, etc.

8.4.2 Health monitoring

Health monitoring is meant to ensure the proper behaviour of the simulation components in the ML sandbox. The sequence diagram is shown in Figure 5.

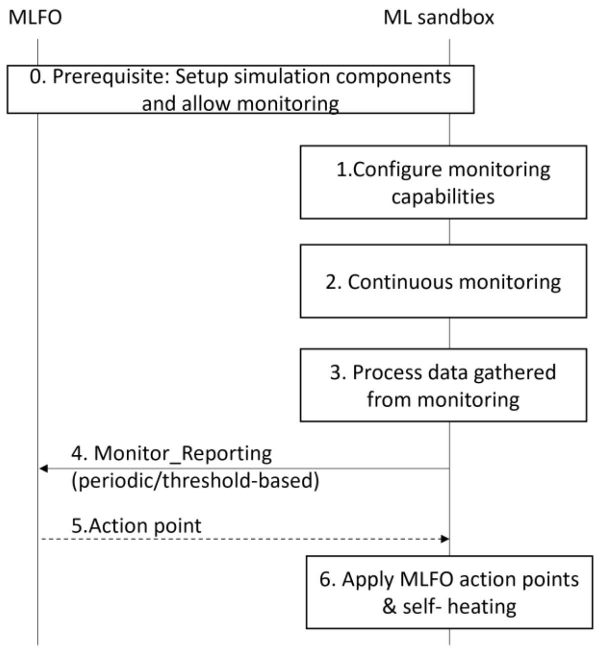
MLFO
ML sandbox
0. Prerequisite: Setup simulation components and allow monitoring
1.Configure monitoring capabilities
2. Continuous monitoring
3. Process data gathered from monitoring
4. Monitor_Reporting (periodic/threshold-based)
5.Action point
6. Apply MLFO action points & self- heating

Figure 5: Health monitoring

Prerequisite**:** Simulation components have been set up and the ML intent allows for monitoring simulation components.

1. ML sandbox sets up the resources for monitoring simulation components.

   NOTE 1 – This step is done by the simulation designer (see clause 8.2.2.1).

2. Monitoring is carried out based on a continuous flow of data generated by simulation components, including simulation data (SRC & SINK nodes), regression tests, reporting from simulation modules, etc.

   NOTE 2 – This step is done by the monitoring agent (see clause 8.2.2.3).

3. Data gathered from monitoring is processed and delivered in the form of reports.

   NOTE 3 – This step is done by the simulation post-processor (see clause 8.2.2.4).

4. Periodic reports are generated and sent to the MLFO, according to ML intent specification. Alternatively, threshold-based alerts can be activated when undesired events occur, which are also reported to MLFO. MLFO may take action after processing periodic reports or threshold-based alerts (see clause 8.4.4) and optionally send action points to the ML sandbox.

5. The ML sandbox applies action points (if any) suggested by MLFO and/or self-healing actions.

   NOTE 5 – The simulator composer may take action for redefining the simulation environment (e.g., switch to a more computation-intensive but accurate tool) if certain indicators of quality are not met.

8.4.3 Validate input/output data

In this scenario, input data and simulated output data are validated. This includes checking the feasibility of training models, the accuracy of generated data, etc. For instance, testing techniques such as equivalence partitioning or centroid positioning can be applied to validate the diversity and the quality of the data generated by simulators.

NOTE 1 – An example of validation of input data is to validate the synthetic data generated by generative adversarial networks (GANs) [b-Castelli-GANs].

The sequence diagram is shown in Figure 6.

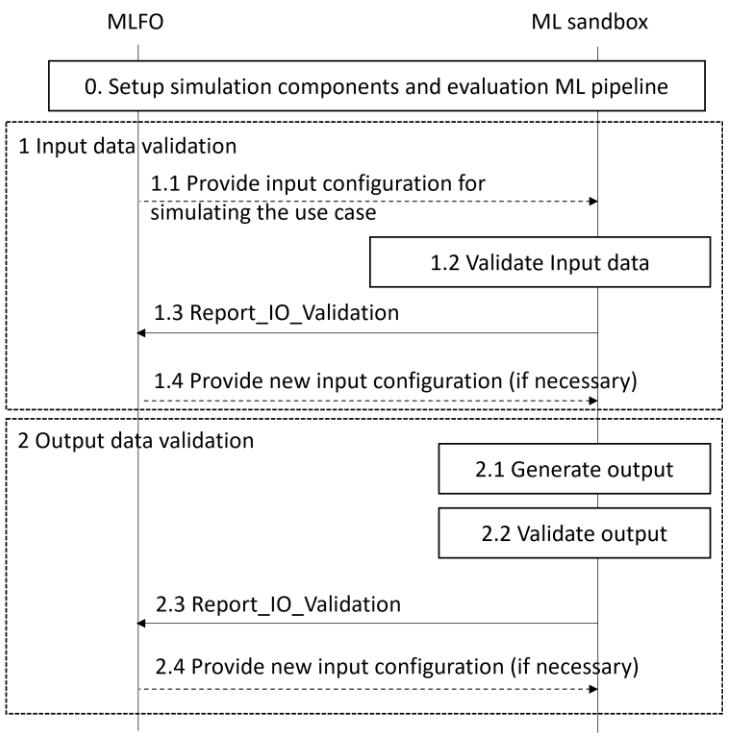
MLFO
ML sandbox
0. Setup simulation components and evaluation ML pipeline
1 Input data validation
1.1 Provide input configuration for simulating the use case
1.2 Validate Input data
1.3 Report_IO_Validation
1.4 Provide new input configuration (if necessary)
2 Output data validation
2.1 Generate output
2.2 Validate output
2.3 Report_IO_Validation
2.4 Provide new input configuration (if necessary)

Figure 6: Validate input/output data

Prerequisite: Simulation components and evaluation ML pipeline have been set up.

For input data:

1.1 ML sandbox receives input configuration from the MLFO to configure input data generation from the simulation environment.

1.2 Input data generation is validated.

NOTE 2 – This step is done by the simulation designer (see clause 8.2.2.1).

1.3 A response with the validation result is provided to MLFO.

1.4 MLFO may provide updated inputs to configure the simulation environment.

For simulated output data:

2.1 ML sandbox receives output data from the ML use case (training data set, trained ML model, evaluation of ML model).

2.2 Output data is validated.

NOTE 3 – This step is done by the simulation post-processor (see clause 8.2.2.4).

2.3 The result of the validation is sent to MLFO.

2.4 MLFO may provide updated inputs to configure the simulation environment.

8.4.4 MLFO-triggered operations

The MLFO-triggered operations are generally defined in Figure 7.

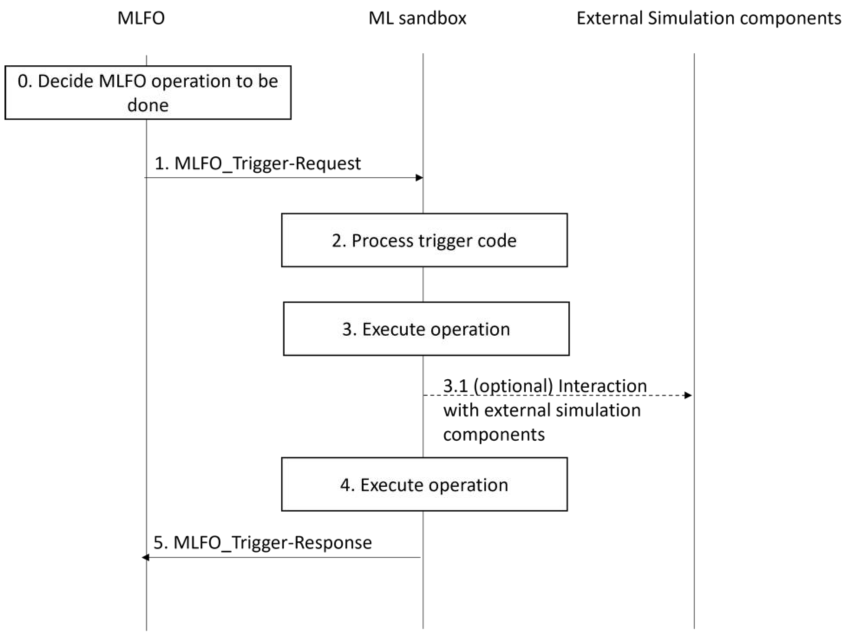
MLFO
ML sandbox
External Simulation components
0. Decide MLFO operation to be done
1. MLFO_Trigger-Request
2. Process trigger code
3. Execute operation
3.1 (optional) Interaction with external simulation components
4. Execute operation
5. MLFO_Trigger-Response

Figure 7: MLFO-trigger operations

The MLFO_Trigger API requests and provides response to the following set of operations:

Setup environment for ML use case (SANDBOX-TRIGGER-001):

Prerequisite: capability discovery is done (see clause 8.2.1).

1. The MLFO sends a request to the ML sandbox to set up the environment for the ML use case in the sandbox. This request contains the selected candidate simulation environment, as a result of the procedure in clause 8.4.1.
2. ML sandbox prepares the environment via the simulation designer (see clause 8.2.2.1):

   a. This step can optionally include download and install of simulation components.
   b. Data handling for the simulated ML underlay includes integration with data brokers and data storage units [ITU-T Y.3174]. A plugin to interact with third party applications can also be employed for the setup.

3. A response is provided by ML sandbox to the MLFO.

   NOTE 1 – Environment setup may use the policies from MLFO and demand mapping (e.g., ML intent may require isolation of resources). The ML underlay may also be configured based on the use case specification and based on features extracted from the live ML underlay to be simulated.

   NOTE 2 – Setting up the evaluation ML pipeline may include the download of ML models from marketplaces [ITU-T Y.3176] and ML model serving [ITU-T Y.3179] to create evaluation ML pipelines.

   NOTE 3 – Environment setup may include the installation of third-party applications or setting up interfaces, establish connections via sockets, etc.

Validate environment for ML use case (SANDBOX-TRIGGER-002):

Prerequisite: Setup environment for ML use case is done (see clause 8.2.2).

1. MLFO requests to run sanity tests on a specific simulation environment for the ML use case.
2. The simulation designer selects and executes the test suite (interaction with third-party applications can be done through the evaluation plugin).
3. The output of the test suite is processed and output to MLFO.

   NOTE 4 – For instance, installing a third-party simulation tool may output a set of traces (or logs) indicating the result of installing submodules. This information should be post-processed to assess that the final installation procedure was successfully accomplished regarding the use case requirement. Example: if the installation of the LTE module failed, but the use case was meant for Wi-Fi (which module was successfully installed), then the result of the installation is satisfactory.

4. The MLFO decides whether the validation results are acceptable or not (may include some basic tests and KPIs).

Manage simulated ML underlay (SANDBOX-TRIGGER-003):

1. The MLFO sends a trigger for modifying/updating the simulated ML underlay. Information on changes is included in the updated ML profile.
2. The simulation driver configures the simulated ML underlay accordingly (e.g., adapt configuration parameters, specify desired output…). Besides, there is an information exchange between the live and simulated ML underlays (mimic purposes).
3. The ML sandbox responds to the trigger with the information related to changes done in the simulated ML underlay (OK/NOK, changelog, etc.).

NOTE 5 – As an example, as a result of network dynamics, some path-loss parameters used by the ML sandbox subsystem may vary over time. To address this issue, the MLFO keeps track of those changes and provides feedback to the Simulation designer to update the necessary simulation parameters.

Evaluate output of ML model (SANDBOX-TRIGGER-004):

Prerequisite: The environment has been set up in ML sandbox for the ML use case.

1. The MLFO sends a request to the simulation composer in the ML sandbox to run the simulated ML underlay.
2. The simulation composer uses the “evaluator plugin” to input the output of the ML model. The plugin can also be used to interact with the third-party application (i.e., translate commands from the MLFO to simulator-oriented instructions).
3. Evaluate the output of an ML model: the evaluation platform provides a report, which is processed by the post-processor.
4. The processed report is sent back to the MLFO. The report can also include synthetic data for training.

NOTE 6 – The information extracted from the simulated ML underlay needs to be post-processed according to the desired output specified in the use case and the characteristics of the simulated ML underlay. The evaluation result can be an OK/NOK message, a percentage of reliability, a set of KPIs gathered from the evaluation procedure, etc.

NOTE 7 – Post-processing for third-party simulation tools may include the conversion of raw data into meaningful information (e.g., average or deviation on KPIs).

ML model training (SANDBOX-TRIGGER-005):

Prerequisite: The ML model is served and evaluation blocks are ready.

1. The MLFO sends a trigger for training an ML model in the sandbox.
2. Model training is performed at the simulated ML underlay through the simulator composer.
3. The trained ML model is post-processed (e.g., compressed, pruned) according to the use case and the policies and capabilities (time constraints, link capacity for exchanging information, storage capabilities, etc.).
4. The trained model is included in the response sent to the MLFO.

Table 8-8 describes the operation codes used in the MLFO-triggered operations.

Table 8-5: Definition of MLFO-triggered operation codes (MLFO trigger types)

MLFO trigger type	Parameters	Time sync / dependencies	Description
SANDBOX-TRIGGER-001	Request type, ML profile	Time sync = yes (evaluation blocks)	Request to prepare the simulation environment (both simulated ML underlay and evaluation ML pipeline) to train, test, and evaluate ML models in the ML sandbox.
SANDBOX-TRIGGER-002	Request type, validation type, acceptance criteria	Time sync = yes (sanity checklist, test suite, etc.)	Request to validate (sanity check) the deployed simulation environment. Used by the MLFO to determine whether to take action to fix potential deployment issues, proceed with ML model evaluation in the sandbox, etc.
SANDBOX-TRIGGER-003	Request type, update type, updated ML profile	Time sync = yes (updated simulation components)	Request to modify/update the simulated ML underlay according to updates in policies, changes in the live ML underlay, potential failures of previous simulated functions, etc.

SANDBOX-TRIGGER-004	Request type, ML profile, ML model output	Time sync = yes (evaluation results)	Request to evaluate the impact of the output of an ML model in the simulated ML underlay, so that some insights can be provided before applying that output to the live ML underlay.
SANDBOX-TRIGGER-005	Request type, ML profile	Time sync = no	Request to train an ML model in the ML sandbox.

8.4.5 Sandbox asynchronous messages

ML sandbox asynchronous messages are generally defined in Figure 8.

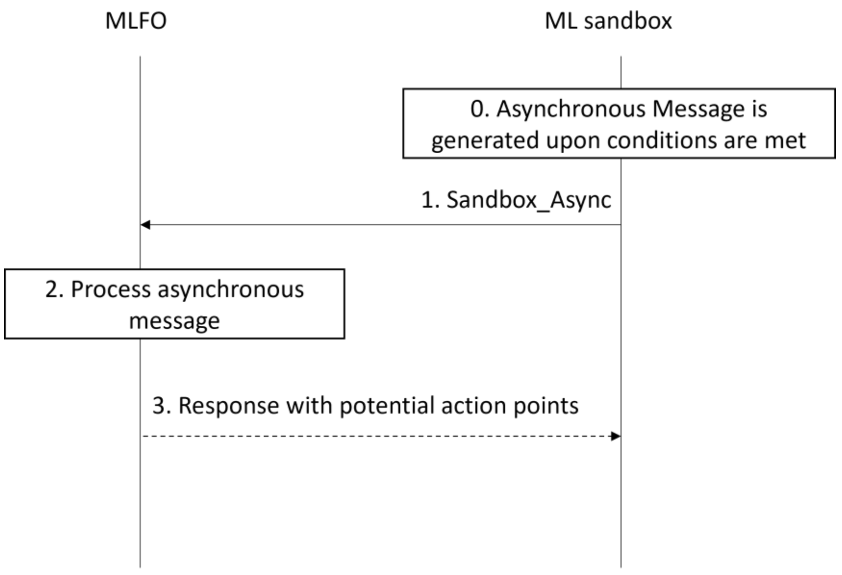
MLFO
ML sandbox
0. Asynchronous Message is generated upon conditions are met
1. Sandbox_Async
2. Process asynchronous message
3. Response with potential action points

Figure 8: ML sandbox asynchronous messages

Prerequisite: Asynchronous messages are generated upon certain conditions are met, which are specific to the different types of events.

1. Upon meeting the trigger conditions, the ML sandbox sends an asynchronous message to the MLFO.
2. The message is processed and potential action points are defined.
3. The MLFO sends a response to the ML sandbox.

The sandbox asynchronous message codes are defined in Table 8-9.

Table 8-6: Definition of ML sandbox asynchronous message codes

Code	Parameters	Conditions	Description
SANDBOX-ASYNC-001	Status code / Error code / detailed report (conditional)	Threshold-based alert is fired / miss-behavior is detected / keep-alive message	Report the health status of the simulation components
SANDBOX-ASYNC-002	Update type / changelog / additional information on implications of update	Updated on simulation components is notified/discovered to/by the ML sandbox	Report an update on security, accounting, licensing requirements of simulation components

SANDBOX-ASYNC-003	Alert type / forecasted results / additional information on potential failure points	Threshold-based alert based on trend analysis (e.g., service at risk)	Proactive behavior trend identification of simulation components and sandbox resources
SANDBOX-ASYNC-004	Report type / Updated simulation component metadata	Change on simulation component is noticed	Report an update on simulation component metadata
SANDBOX-ASYNC-005	Report type / Updated level of intelligence of simulation components	Change on the intelligence level of a simulation component	Report the simulation environment intelligence level

9. Security considerations

This Recommendation describes an architectural framework for the ML sandbox in the context of integrating machine learning 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 unauthorized access to, and data leaking from, the ML sandbox, whether or not there is a malicious intention, with the implementation of appropriate mechanisms, such as those for authentication and authorization, and external attack protection.

Bibliography

[b-JSON] IETF RFC 8259 (2017), *The JavaScript Object Notation (JSON) Data Interchange Format (JSON)*

[b-CSV] IETF RFC 4180 (2005), *Common Format and MIME Type for Comma-Separated Values (CSV) Files*

[b-XML] W3C Recommendation (2008), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*

[b-ITU-T Y.Sup55] ITU-T Supplement ITU-T Y.3170-series (2019) "*Machine learning in future networks including IMT-2020: use cases"*

[b-Riley-ns3] Riley, G. F., & Henderson, T. R. (2010). *The ns-3 network simulator.* In Modeling and tools for network simulation (pp. 15-34). Springer, Berlin, Heidelberg.

[b-IEEE 802.11] IEEE 802.11 WLANs. The Working Group for WLAN Standards, 2015.

[b-Zhang-Testing] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). *Machine learning testing: Survey, landscapes and horizons.* IEEE Transactions on Software Engineering.

[b-Castelli-GANs] Castelli, M., Manzoni, L., Espindola, T., Popovič, A., & De Lorenzo, A. (2021). *Generative adversarial networks for generating synthetic features for Wi-Fi signal quality*. PloS one, 16(11), e0260308.