BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **22**, No 3
Sofia • 2022 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2022-0034

Toward Programmability of Radio Resource Control Based on O-RAN

Evelina N. Pencheva[1]*, Ivaylo I. Atanasov*[2]

[1]*Todor Kableshkov University of Transport, Geo Milev St., 158, 1113 Sofia, Bulgaria*
[2]*Technical University of Sofia, St. Kliment Ohridski blvd, 8, 1756 Sofia, Bulgaria*
E-mails: evelina.nik.pencheva@gmail.com iia@tu-sofia.bg

Abstract**:** *Open Radio Access Network* (*O-RAN*) *is a concept that aims at embedding intelligence at the network edge and at disaggregating of network functionality from the hardware. The paper studies how the O-RAN concept can be used for optimization of radio resource management. The research focuses on adaptive radio resource allocation based on predictions of device activity. For narrowband devices which send sporadically small volumes of data, a feature is defined which enables a device with no activity for a short time to suspend its session and to resume it moving in active state. Dynamic configuration of the inactivity timer based on prediction of device activity may further optimize radio resource allocation. The paper studies an O-RAN use case for dynamic radio resource control and presents the results of emulation of the RESTful interface defined between the O-RAN non-real-time and near real-time functions.*

Keywords**:** *Mobile communications, Radio Access Network, Radio Resource Management, Adaptive control, Policy management.*

1. Introduction

The Open Radio Access Network (O-RAN) initiative applies concepts of agility, virtualization, and intelligence, and it is aimed at openness of RAN architecture to provide intelligent radio control toward and beyond 6G networks [1]. The increasing complexity of the radio networks calls for embedding intelligence in every layer of RAN architecture to provide dynamic radio resource allocations and optimization of network efficiency [2, 3]. The O-RAN concept disaggregates the RAN hardware and software applying both horizontal split (centralized unit, distributed unit, and radio unit) and vertical split (control plane and user plane separation). The vertical split follows the principle of Software Defined Networking (SDN) and enables separate optimization of control plane and user plane function, fosters interoperability in multi-vendor environment enabling consistent control on network entities. The horizontal disaggregation aims performance gains from centralized resource management and joint multicell signal processing, dynamic configuration and localization of virtualized network functions based on the requirements of different use cases etc. RAN openness speeds up innovations and promises cost reduction. O-RAN enables cloud computing at the edge to implement complex network functions and to build artificial intelligence for improving radio resource control and management and to provide better RAN performance [4-6].

The O-RAN specifications define both non-real-time services and near-Real-Time (near-RT) services [7]. The non-Real-Time (non-RT) services are provided by the Service Management and Orchestration (SMO) layer of O-RAN logical architecture with the purpose to optimize RAN performance using Machine Learning (ML) models and other Artificial Intelligence (AI) applications. The non-RT policy management service provides to the near RT functions, policy-based guidance containing statements on policy resources and policy objectives. The non-RT enrichment information service provides information in addition to generally available one to a network entity to improve its performance. The non-RT ML model management service enables training and execution of ML models at different places in the O-RAN architecture. The near-RT services provide means to control the RAN nodes functionality for Radio Resource Management (RRM), e.g., the near-RT RIC may monitor, suspend, stop, override, or control the behaviour of RAN nodes via policies

Several use cases related to RAN optimization and control within O-RAN are defined [8]. In this paper, we study another use case of O-RAN function deployment for RRM optimization. It is related with adaptive control of one of the new features in the New Radio (NR) design.

The growing number of devices and the requirements of diverse use case in next generation networks calls for solutions that decrease the signalling load to increase overall system capacity, latency, and battery consumption. The reduced signalling results in increase of system capacity also. One of the performance improvements in NR design to support narrowband Internet of Things (IoT) relates to introduction of a new device state called inactive state in addition to idle state with no data transmission and connected state with data exchange. The new state is defined mainly for narrowband IoT devices which usually exchange with the network small amount of data, which are not enough urgent to justify the energy required to move from idle to connected state. The device is in inactive state for a short time after transmission inactivity. In this state, the radio configuration and security parameters are stored to enable fast transition to connected state. The key benefits of the inactive state are significant reduction in latency and minimization of battery consumption due to decreased signalling. It is RAN that configures statically the duration of the maximal period to stay in inactive state in the respective RAN slice based on the use case requirements. However, this static configuration may not be optimal in unforeseen and emergency situations. Dynamic control of the inactivity timer duration may be applied based on device activity and here is the role of embedded RAN intelligence.

The paper is structured as follows. Next section argues the research motivation. Section 3 presents the intelligent O-RAN architecture and the respective control loops and services. Section 4 provides a detailed description of the considered use case and discusses how non-RT services and near-RT services may be used to enable more intelligent handling of device inactivity state. Section 5 presents results of emulation of the REST-based interface between O-RAN nonreal-time and near real-time functions. Finally, Section 6 concludes the paper and summarizes the contribution.

2. Research motivation

Radio Resource Control (RRC) protocol operates between User Equipment (UE) and the base station (gNB) and plays an important role in communications between them. Fig. 1 shows the RRC state machine as define in [9].

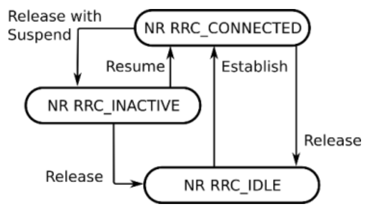

Fig. 1. RRC state machine in 5G [9]

In RRC_idle state, the UE context is not stored neither at the UE nor in RAN. No data transfer is possible as the UE sleeps most of the time and thus reduces the battery consumption. The UE wakes up periodically to listen for paging. The transition to RRC_Connected state occurs when the UE initiates RRC connection setup to receive call messages and notifications of public warning systems. In RRC_Connected state, the UE context is stored at the UE and RAN, and the UE can transfer data and then discontinues reception to configure to reduce the power consumption. In RRC_Inactive state, if the UE temporary is not involved in data transfer, the UE context is stored in both the UE and RAN. The UE location is known in RAN registration area and the UE may move without needing to notify the network which reduces the UE power and RAN resource consumption. Also, the UE is allowed to sleep in RRC_Inactive state and the mobility is controlled by the UE through cell reselection without network involvement.

The benefits of RRC connection control for energy efficiency enhancement in NR are analysed in [10-12]. The research in the area shows that the introduction of RRC_Inactive state may reduce the overhead over 29%, the delays by 12% and the power consumption over 21% compared to LTE [13]. The configuration of RRC_Inactive state for machine type communications may reduce more than 200% latency compared to RRC_Idle state and up to 40% UE power consumption compared to RRC_Connected state.

RRC state transitions can be configured statically and optimized for different service requirements in RAN slices [14, 15]. However intelligent RRC control requires more flexible RRC state transitions based on dynamic radio conditions, current cell loading and prediction about UE activity. When the UE is in RRC_Connected state, it is the network that may initiate RRC connection suspension or RRC connection release. The network decision is based on UE inactivity and the configured parameters according to the appropriate for the respective use case slice. It should be stressed that many service envisioned in 5G and beyond cannot easily be mapped onto the three 5G main service types. Some examples include augmented reality which has requirements for both low latency and high bandwidth, and factory of the future where both energy efficiency and low latency are valuable. As these services combine challenges and requirements related to multiple service types, the static configuration of RAN slices may be difficult and, in some cases, even impossible. This calls for dynamic RRC state transition configuration based intelligent prediction of UE transmission dynamics.

The aim of our research is to study how the O-RAN concept may be used to provide intelligent and flexible RRC state transitions to optimize the network performance. The DataInactivity timer indicates how long the UE context and established radio bearers must be stored during UE inactivity. Typically, the DataInactivity timer is statically configured in the RAN slider with value between few seconds to a few tens of seconds [16]. With the deployment of O-RAN, this timer may be dynamically controlled in case of e. g. emergency situations. The brief description of the use case related to RRC state transition control is as follows.

The non-RT services may collect enrichedent information about activity of a given group of UEs in RAN. This information and necessary measurement metrics reported by the network level may be used to construct/train AI/ML models. Based on reasoning about UE activity, RRC state machine policies may be generated and send to the near-RT services for enforcement and evaluation. The near-RT service, in turn, may interpret the received policies and determine the required changes to optimize the RRC state transitions. To obtain network performance that fulfils the policy criteria, the near-RT services may trigger the procedures in RAN nodes. So, the research motivation is to allow operators to flexibly configure the desired optimization policy, utilizing the right performance criteria and leveraging AI/ML to predict UE activity with the aim to enable intelligent RRC state transitions.

3. O-RAN architectural considerations

Traditional RAN consists of radio and baseband units, where the interface between them is standardized. However, contemporary products are implemented in proprietary variations which make difficult the interoperability between multiple vendors.

The programmable O-RAN architecture accommodates use cases with different QoS requirements. The disaggregation of hardware and software allows flexibility and scaling in comparison with integrated platforms. The introduction of additional interfaces facilitates introduction of advanced RAN features and capabilities. Both network function virtualization (NFV) of the centralized baseband unit and the standardized open RAN interfaces enable efficient RRM that fits to the diverse application requirements using network slicing.

Fig. 2 shows the logical O-RAN architecture as defined in [7].

3GPP RAN specifications define gNB Centralized Unit (gNB-CU) as a logical unit which hosts higher level RAN protocols, and gNB Distributed Unit (gNB-DU) handling low layer RAN protocols. In O-RAN architecture, the following logical nodes are defined:

- O-RAN Central Unit (O-CU) is a logical node dealing with the radio resource control, packet data convergence and service data adaptation. It is responsible for user session management e. g. handover, load balancing etc. The vertical splitting into O-CU-Control Plane (O-CU-CP) and O-CU-User Plane (O-CU-UP) is logically equivalent to 3GPP CU vision.
- O-RAN Distributed Unit (O-DU) supports function of low level protocol and is responsible for both the management of medium access like RAN slicing, scheduling policy etc., and the radio management like resources scheduling.
- O-RAN Radio Unit (O-RU) hosts the low physical layer functions including RF processing. It is responsible for device management like modulation, blockage detection, interference control, etc.

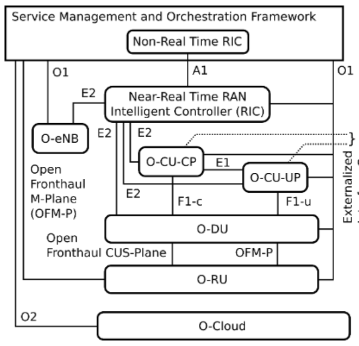
Service Management and Orchestration Framework
Non-Real Time RIC
O1
A1
O1
E2
Near-Real Time RAN Intelligent Controller (RIC)
O-eNB
E2
E2
O-CU-CP
E1
O-CU-UP
Open Fronthaul M-Plane (OFM-P)
E2
F1-c
F1-u
Externalized Interface Groups
O-DU
Open Fronthaul CUS-Plane
OFM-P
O-RU
O2
O-Cloud

Fig. 2. Logical O-RAN architecture [7]

O-Cloud provides the physical infrastructure for cloud computing and NFV to manage nodes and virtualized functions deployment.

Service Management and Orchestration (SMO) framework is a platform that manages the lifecycle of network functions and the O-Cloud. It includes an environment for fast development of application, support for access control and AI/ML lifecycle management, service orchestration, policy control, and topology, allowing portability of applications.

The O-RAN architecture introduces a new concept of RAN Intelligent Controller (RIC) which provides a centralized abstraction of control plane functions enabling intelligent handling of RAN resources. The RIC centralizes the intelligence in the closed control loops for both non-RT and near-RT functions allowing operators to optimize RAN through gathering data from RAN nodes and issuing autonomous actions. The disaggregation into near-RT RIC and non-RT RIC functionality meets the timing requirements of operations from one millisecond for real-time control to thousands of milliseconds for traffic forecasting and slice allocation. The near-RT RIC hosts third party applications (xApps) which use open APIs to provide data-driven intelligent control. The non-RT RIC performs operations such as training of AI/ML models.

The key management interfaces defined in the O-RAN architecture are as follows:

- A1 interface is defined between non-RT RIC in SMO framework and near-RT RIC. It is used to provide the near-RT RIC with enrichment information, policies, and ML model updates, and to provide feedback information to the non-RT RIC how the policy set works.
- E2 interface is defined between near-RT RIC and an E2 node such as O-CU-CP, O-CU-UP, O-DU or any combination of them. It uses messages to monitor, suspend, override, and control the E2 nodes, executes actions coming from xApps or near-RT RIC, and gets data collection from E2 nodes.
- O1 interface is defined between SMO framework and the O-RAN managed elements. O1 for O-CU-CP and O-CU-UP supports installation, software management, configuration management, performance management, fault management and file management.
- O2 interface is defined between SMO framework and the O-Cloud. It enables O-Cloud infrastructure management and management of the cloudified network functions that run on the O-Cloud.
- Open fronthaul Management plane (M-plane) is defined to control the O-RU by SMO or DU. It supports installation, software management, configuration management, performance management, fault management and file management of RU.

The A1 interface is used by O-RAN non-RT control loop when it is involved in coordination between non-RT and near-RT RICs. The control loop is responsible for management of the resource orchestration by applying policies after the respective decisions. It uses interference models and optimization algorithms based on processing data from many sources. Typical data-driven non-RT control includes slice management and selection of inference models which should be executed. The control loop decisions can be made according to, e.g., service level agreements, computational resources, data availability, etc.

The O-RAN near-RT control loops run between near-RT RIC and O-CU-CP, O-CU-UP and O-DU through E2 interface. These control loops use ML-based algorithms, implemented as xApps on the near-RT RIC to make intelligent decisions about resource allocation for optimal quality of experience, scheduling, handover procedures, load balancing, etc. The E2 functions include near-RT RIC services, and near-RT RIC support functions for interface management and near-RT RIC service update. The O-RAN RT control loops concern interaction between DU elements, and between DU and RU. These loops are not currently defined in O-RAN specifications as ML models are not supported at the DU. Possible data-driven decisions may be made for modulation and coding schemes, interference reduction, etc.

The transport network layer of the A1 interface is built on IP transport. HTTP on the top of TCP/IP provides reliable transport. The RESTful approach is used on the application layer to transfer policy statements in JSON format.

Next section provides more detailed description of the entities and resources involved in the RRC state transition control, the possible solution, and the required data.

4. User activity-based resource optimization

UE activity-based resource optimization may be used when the RAN has been configured to save the UE context and established radio bearers to wait for some time after some period of UE inactivity for certain users. One such scenario is when the RAN has been configured to support RAN slices. The desired RAN behaviour for slices is configured over O1 interface. The DataInactivity timer may be configured with different wait time values for different UEs. When a RAN slice is instantiated the RAN functionality is configured accordingly. In such way, the RAN behaviour can fulfil the slice for most situations. However, cases and places exist where the RAN resources are not enough to fulfil the slice which performance is continuously monitored by SMO. In these cases, the A1 policies provided by the non-RT RIC can be used to improve the situation. To determine the right A1 policy the non-RT RIC uses additional information available in SMO.

Take an e-Health service as an example of slice tenant. In the context of an e-health service, a set of personal sensors provide the information flows to make the vital signs monitoring possible. The sensors are configured to send measurements periodically or upon approaching predefined triggers. In normal situation, the DataInactivity timer is configured with predefined value in the respective slice. This value is a result of rigorous planning based on Service Level Agreement (SLA). The SMO monitors fulfilment of the respective slice parameters and xAPP applications collect data for sensors' activities. In pre-emergency conditions, the sensors may increase the frequency and volume of messages submitted and it is necessary to take actions. The non-RT RIC may use the intelligent decisions of xAPP about expected sensors activity and may influence on the DataInactivity timer value by extending the wait time through an A1 policy based on prediction of UE activities in similar situations. In this case, the duration of the DataInactivity timer is extended.

The entities involved in the use case include SMO including non-RT RIC, near-RT RIC, RAN and Application Server. The non-RT RIC monitors the fulfilment of slice metrics to construct/update the relevant AI/ML model that will be deployed in the near-RT RIC to assist the RRC state transition control function. It is also responsible to train the potential ML models to predict the activity of a UE or a group of UEs, and to send policies to the near-RT RIC to drive the RRC state transition in terms of expected activity. The near-RT RIC supports the execution of the AI/ML models from non-RT RIC, interprets and executes the provided policies, and sends performance metrics to the non-RT RIC for evaluation and optimization. The RAN enforcement of RRC state transition control is based on A1/E2 messages, which are expected to influence the behaviour of RRM functions. The Application Server supports data collection with required granularity from UE(s) and communication of real-time traffic related data about the monitored UE(s) as enrichment data. The flow for the use case consists of the following steps, shown in Fig. 3.

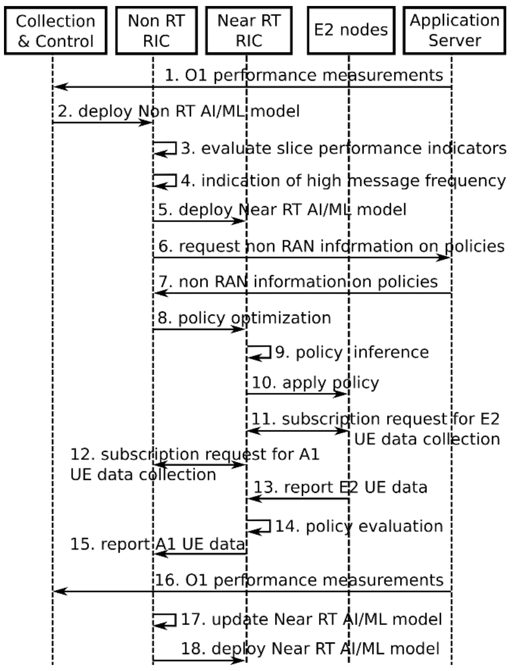
Collection & Control
Non RT RIC
Near RT RIC
E2 nodes
Application Server
1. O1 performance measurements
2. deploy Non RT AI/ML model
3. evaluate slice performance indicators
4. indication of high message frequency
5. deploy Near RT AI/ML model
6. request non RAN information on policies
7. non RAN information on policies
8. policy optimization
9. policy inference
10. apply policy
11. subscription request for E2 UE data collection
12. subscription request for A1 UE data collection
13. report E2 UE data
14. policy evaluation
15. report A1 UE data
16. O1 performance measurements
17. update Near RT AI/ML model
18. deploy Near RT AI/ML model

Fig. 3. RRC state transition control flow diagram: policy generation and evaluation

1. As a precondition, the SMO has established data collection and the non-RT RIC has access to this data.

2. Non RT AI/ML models are deployed.

3. The non-RT RIC analyses the historical data from RAN and Application server for training the relevant AI/ML model required to predict the UE(s) activity and to define the policy for real time optimization of configuration.

4. An event or operator specified trigger condition occurs.

5. The non-RT RIC deploys the AI/ML model in the near-RT RIC via O1 interfaces or it assigns the AI/ML model for the xApp via A1. This model is required to real time resource usage optimization.

6. The non-RT RIC requests the Application Server to provide non RAN information on policies.

7. The Application server provides the requested non RAN information to the non-RT RIC.

8. The non-RT RIC provides the relevant policy and enrichment data to the near-RT RIC over A1 interface. The enrichment data may include information about message frequency, session duration data, inactivity duration data, UE location, etc.

9. The near-RT RIC deduces the optimal DataInactivity timer value to be configured according to the trained AI/ML model.

10. The near-RT RIC enforces the result to the RAN via E2 interface.

11. The non-RT RIC configures specific performance measurement data to be collected from RAN. These data can be used to assess the performance of the RRC state transition control function in near-RT RIC, or to evaluate the result of the applied policy and configuration.

12. The near-RT RIC subscribes for UE data with the E2 nodes and requests UE data collection.

13. The near-RT RIC receives reports from the RAN using E2 interface.

14. The near-RT RIC evaluates the policy.

15. The non-RT RIC receives UE data reports via A1 interface.

16. The Collect&Control function in the SMO receives O1 performance measurements.

17. The non-RT RIC updates the near-RT AI/ML model.

18. The updated near-RT AI/ML model is deployed into the near-RT RIC.

The Steps 16-18 are repeated until an operator specified trigger condition or event is satisfied.

Multi-dimensional data need to be reported to non-RT RIC to train the AI/ML model and for policy/intent generation. The non-RT RIC should monitor enrichment information about resource consumption in the area to detect approaching thresholds that the configured RAN parameters will not be enough to fulfil the requirements. At this point, the non-RT RIC needs to configure more detailed information to be reported for the UE(s) to have better insight in performance, e.g., the frequency of UE messages and the duration of UE inactivity periods. The external enrichment information is required for the SMO, e.g., QoS related measurements.

The considered use case can be addressed to any other unusual or unforeseen situations, where the activity pattern of narrowband devices changes or in case of unexpected traffic load or bad radio conditions. Examples where an extension of the inactivity timer is required due to transmission of more than usually data include monitoring of the driver's general mode and depression symptoms during car driving or monitoring of trackside environment in case of danger of rock mass collapses. Situation in which a reduction of the inactivity timer is required include, e.g., cell congestion due to some emergency circumstances.

The A1 interface is used to provide policy management service and enrichment information service. The non-RT RIC uses the policy management function to provision and manage A1 policies that should be enforced by the near-RT RIC. The enrichment information service provides information along with the generally available information from one or more sources. SMO collects information from both O-RAN internal and O-RAN external sources.

The A1 protocol definition is based on REST (REpresentational State Transfer) architectural style. A policy is defined as a resource which can be manipulated by HTTP methods. A particular policy URI can be found in the policy resource tree as defined in [17].

The protocol implementation requires definition of policy state machine in the non-RT RIC and in the near-RT RIC.

A policy lifecycle state machine as seen by the non-RT RIC is proposed and shown in Fig. 4.

The policy state machine as seen by the near-RT RIC is shown in Fig.5. From the non-RT RIC point of view, a policy must be created, updated, or deleted when predefined measurement thresholds are reached. The non-RT RIC requests enforcement of a newly created policy or updated policy, or deletion of a policy.

Upon receiving of a request from the non-RT RIC, the near-RT RIC tries to enforce the instructions via E2 interface and sends back the result.

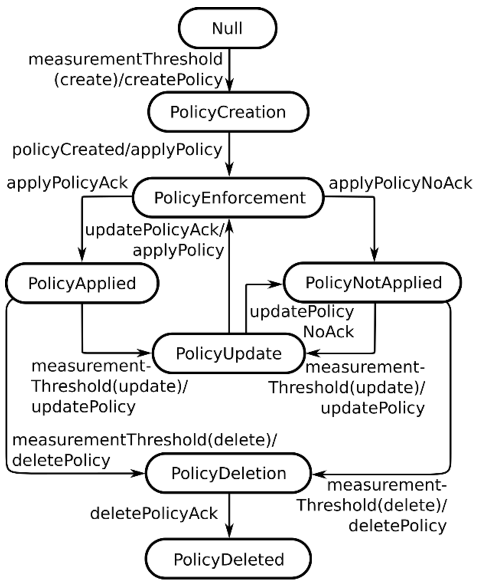
Null
measurementThreshold (create)/createPolicy
PolicyCreation
policyCreated/applyPolicy
applyPolicyAck
PolicyEnforcement
applyPolicyNoAck
updatePolicyAck/ applyPolicy
PolicyApplied
PolicyNotApplied
updatePolicy NoAck
PolicyUpdate
measurement-Threshold(update)/ updatePolicy
measurement-Threshold(update)/ updatePolicy
measurementThreshold(delete)/ deletePolicy
PolicyDeletion
measurement-Threshold(delete)/ deletePolicy
deletePolicyAck
PolicyDeleted

Fig. 4. The policy life cycle state machine – the non-RT RIC view

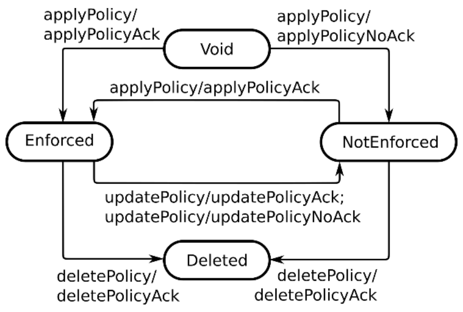
applyPolicy/ applyPolicyAck
Void
applyPolicy/ applyPolicyNoAck
applyPolicy/applyPolicyAck
Enforced
NotEnforced
updatePolicy/updatePolicyAck; updatePolicy/updatePolicyNoAck
Deleted
deletePolicy/ deletePolicyAck
deletePolicy/ deletePolicyAck

Fig. 5. The policy life cycle state machine – the near-RT RIC view

The near-RT services and the related E2 procedures include REPORT, INSERT, CONTROL and POLICY. The REPORT service enables subscription to and notification about events related to procedures in the centralized units or distributed units. The INSERT service is used to suspend the associated procedure after event occurrence. The CONTROL service enables initiation of a new associated procedure or to resume a previously suspended one. The POLICY service is used to request applying a specific policy after occurrence of specific event. For brevity, the E2 procedures are not shown in Fig. 5.

Both state machines must be synchronized. To prove that both state machines expose equivalent behavior, we describe the state machines as Labelled Transition Systems (LTSs) and use the mathematical formalism of weak bi-simulation.

An LTS is a quadruple of a set of states, a set of labels (regarded as actions that drive the transitions), a set of transitions, and a set of initial states [18].

By $T_{nonRT} = \{S_{nonRT}, A_{nonRT}, \rightarrow_{nonRT}, s_{nonRT}\}$ is denoted an LTS, representing the policy life cycle state machine supported by the non-RT RIC, where:

S_{nonRT} = {Null [s^{non}_1], PolicyCreation [s^{non}_2], PolicyEnforcement [s^{non}_3], PolicyApplied [s^{non}_4], PolicyNotApplied [s^{non}_5], PolicyUpdate [s^{non}_6], PolicyDeletion [s^{non}_7], PolicyDeleted [s^{non}_8]}.

A_{nonRT} = {measurementThreshould(create) [a^{non}_1], policyCreated [a^{non}_2], applyPolicyAck [a^{non}_3], applyPolicyNoAck [a^{non}_4], measurementThreshould(update) [a^{non}_5], updatePolicyAck [a^{non}_6], updatePolicyNoAck [a^{non}_7], measurementThreshould (delete) [a^{non}_8], deletePolicyAck [a^{non}_9]}.

$\rightarrow_{nonRT} = \{(s^{non}_1\ a^{non}_1\ s^{non}_2), (s^{non}_2\ a^{non}_2\ s^{non}_3), (s^{non}_3\ a^{non}_3\ s^{non}_4), (s^{non}_3\ a^{non}_4\ s^{non}_5), (s^{non}_4\ a^{non}_5\ s^{non}_6), (s^{non}_5\ a^{non}_5\ s^{non}_6), (s^{non}_6\ a^{non}_7\ s^{non}_5), (s^{non}_6\ a^{non}_6\ s^{non}_3), (s^{non}_4\ a^{non}_8\ s^{non}_7), (s^{non}_5\ a^{non}_8\ s^{non}_7), (s^{non}_7\ a^{non}_9\ s^{non}_8)\}$.

$s_{nonRT} = s^{non}_1$.

Short notations for the names of states, actions and transitions are given in brackets.

By $T_{nearRT} = \{S_{nearRT}, A_{nearRT}, \rightarrow_{neaRT}, s_{nearRT}\}$ is denoted an LTS, representing the policy state machine supported by the near-RT RIC, where:

S_{nearRT} = {Void [s^{near}_1], Enforced [s^{near}_2], NotEnforced[s^{near}_3], Deleted [s^{near}_4]}.

A_{nearRT} = {applyPolicy [a^{near}_1], updatePolicy [a^{near}_2], deletePolicy [a^{near}_3]}.

$\rightarrow_{nearRT} = \{(s^{near}_1\ a^{near}_1\ s^{near}_2), (s^{near}_1\ a^{near}_1\ s^{near}_3), (s^{near}_2\ a^{near}_2\ s^{near}_3), (s^{near}_3\ a^{near}_2\ s^{near}_3), (s^{near}_3\ a^{near}_1\ s^{near}_3), (s^{near}_3\ a^{near}_1\ s^{near}_2), (s^{near}_2\ a^{near}_3\ s^{near}_4), (s^{near}_3\ a^{near}_3\ s^{near}_4)\}$.

$s_{nearRT} = s^{near}_1$.

Bi-simulation is a binary relationship between the states of two LTS which associates the LTS behaviour as equivalent, i.e., one LTS simulates the other LTS and vice versa. The concept is used to prove the behavioural equivalence of concurrent processes [19]. While the strong bi-simulation requires a strict correspondence between the states of the two LTSs, the weak bi-simulation means that there may be internal actions which are not visible for external observers.

Proposition. T_{nonRT} and T_{nearRT} have a weak bi-simulation relationship.

Proof: By R is denoted a relationship between the states of T_{nonRT} and T_{nearRT} where:

$R = \{(s^{non}_1, s^{near}_1), (s^{non}_4, s^{near}_2), (s^{non}_5, s^{near}_3), (s^{non}_8, s^{near}_4)\}$.

To prove the existence of a weak bi-simulation between T_{nonRT} and T_{nearRT} , it is necessary to show that all transitions from states in a couple in R terminate into states in a couple of R.

The following transition mapping may be identified:

1. The non-RT RIC creates a new policy, requests the policy enforcement and the near-RT RIC responds with policy enforcement acknowledgement: for $\forall$ $(s^{non}_1\ a^{non}_1\ s^{non}_2), (s^{non}_2\ a^{non}_2\ s^{non}_3), (s^{non}_3\ a^{non}_3\ s^{non}_4)\ \exists\ (s^{near}_1\ a^{near}_1\ s^{near}_2)$.

2. The non-RT RIC creates a new policy, requests the policy enforcement and the near-RT RIC does not acknowledge the policy enforcement: for $\forall$ $(s^{non}_1\ a^{non}_1\ s^{non}_2), (s^{non}_2\ a^{non}_2\ s^{non}_3), (s^{non}_3\ a^{non}_4\ s^{non}_5)\ \exists\ (s^{near}_1\ a^{near}_1\ s^{near}_3)$.

3. The non-RT RIC updates a policy that has been enforced, requests the updated policy enforcement and the near-RT RIC acknowledges the policy

enforcement: for $\forall$ (s^{non}_4 a^{non}_5 s^{non}_6), (s^{non}_6 a^{non}_6 s^{non}_3), (s^{non}_3 a^{non}_3 s^{non}_4) $\exists$ (s^{near}_2 a^{near}_2 s^{near}_3), (s^{near}_3 a^{near}_1 s^{near}_2).

4. The non-RT RIC updates a policy that has not been enforced, requests the updated policy enforcement and the near-RT RIC acknowledges the policy enforcement: for $\forall$ (s^{non}_5 a^{non}_5 s^{non}_6), (s^{non}_6 a^{non}_6 s^{non}_3), (s^{non}_3 a^{non}_3 s^{non}_4) $\exists$ (s^{near}_3 a^{near}_2 s^{near}_3), (s^{near}_3 a^{near}_1 s^{near}_2).

5. The non-RT RIC requests an update of a policy that has been enforced and the near-RT RIC does not acknowledge the policy update: for $\forall$ (s^{non}_4 a^{non}_5 s^{non}_6), (s^{non}_6 a^{non}_7 s^{non}_4) $\exists$ (s^{near}_2 a^{near}_2 s^{near}_3).

6. The non-RT RIC requests an update of a policy that has not been enforced and the near-RT RIC does not acknowledge the policy update: for $\forall$ (s^{non}_5 a^{non}_5 s^{non}_6), (s^{non}_6 a^{non}_7 s^{non}_5) $\exists$ (s^{near}_3 a^{near}_2 s^{near}_3).

7. The non-RT RIC deletes an enforced policy: for $\forall$ (s^{non}_4 a^{non}_8 s^{non}_7), (s^{non}_7 a^{non}_9 s^{non}_8) $\exists$ (s^{near}_2 a^{near}_3 s^{near}_4).

8. The non-RT RIC deletes a policy that is not enforced: for $\forall$ (s^{non}_5 a^{non}_8 s^{non}_7), (s^{non}_7 a^{non}_9 s^{non}_8) $\exists$ (s^{near}_3 a^{near}_3 s^{near}_4).

Therefore, T_{nonRT} and T_{nearRT} have a weak bi-simulation relationship, i.e., they expose equivalent behaviour. ■

The non-RT RIC requests a new policy enforcement by using an HTTP POST method, an updated policy enforcement by using an HTTP PUT method and a policy deletion by using an HTTP DELETE method. The HTTP method is used to retrieve the policy status. The payload in HTTP procedures represents an A1 policy in JSON format. Each policy has an identifier and belongs to a certain policy type. The policy type identifier is constructed by the policy type name and version.

An example of JSON description of A1 policy for adaptive RRC control is given below.

```
{
  "group_id": "1",
  "ue_id_list": ["1","2","3"],
  "policy_id": "1",
  "scope": {
    "group_id": "1",
    "cell_id": "A"
  },
  "statement": {
    "PDU_Session_InactivityTimer":60,
    "DRB_InactivityTimer":60
  }
}
```

The policy is for a group of three stationary UEs with user traffic, which corresponds to 5 G Quality of service Identifier (5QI) = 8. The UEs are in cell A. For this group of UEs, the data inactivity periods for Packet Data Unit (PDU) Session Inactivity Timer and for Data Radio Bearer (DRB) Inactivity Timer are set to 60 s.

5. Emulation of policy management service

In the considered scenario, the A1 interface is used by the non-RT-RIC to send policies to the near-RT-RIC related to the management and control on the dynamic change and maintenance of the RRC state. The purpose is to improve the RAN performance.

The policy may be of imperative or declarative type. While the declarative policy uses statements to express the policy goal, but not how to accomplish it, the policy of imperative type uses statements to explicitly change the of the managed objects.

In different mission-critical application scenario where the failure or interruption of the system may lead to significant impacts and may have catastrophic consequences for people and infrastructure, the accurate and timely provision of appropriate instructions is essential. Dedicated networks with distributed core functionality can be deployed for such mission critical businesses, e.g., for the railways or for healthcare applications.

An example is an intelligent application for trackside monitoring in railways which receives track performance data from devices equipped with sensors and configured to send periodically or triggered reports. The devices need to camp in low activity state, they wake up seldom to transmit and receive a relatively small payload. In case of critical situation, when the application discovers some context specific exceeding critical threshold values, the gNB may extend the inactivity timer to enable uplink data transfer. More frequent data transfer can be required for decision about the train speed, which can be implemented as a policy. Using ML models for predicting device activity, the non-RT RIC can influence the adaptive configuration of inactivity timer for a selected device(s) through a statement in an imperative A1 policy. Using the A1 policy, the RAN can assure the dynamically defined inactivity time periods correspond to the device needs for data transmission.

For mission critical applications, latency is an essential key performance indicator. For such applications, the A1 policy management functionality makes the interface more special to more genetic RESTful services. Latency refers to time intervals which quantifies the delay between an event and the respective target effects. In the context of A1 interface, the Round-Trip Time (RTT) is measured for the control plane as the time taken for a request (e.g., for installing an imperative A1 policy) generated by the non-RT RIC to go to the destination, be replied and travel back. This RTT definition assumes that response time is supposed to be variable, but the RTT does not depend on the computational load. RTT can change over time for the same station and usually it is described by a RTT profile over time. RTT statistics is summarized through the maximum, mean and minimum value of RTT, the variance, the value of a given percentile, etc.

That is the reason to evaluate the injected RTT (referred as latency) by the A1 interface.

To emulate the A1 policy management service an experiment is setup. The RESTful approach, that is adopted, imposes the well-known client-server pattern, and the setup for the numerical experiment is shown in Fig. 6.

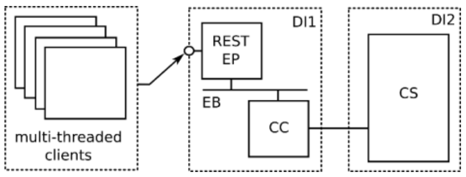

Fig. 6. Numerical experiment setup for A1 policy management service endpoint's latency estimation

The RESTful EndPoint (EP), exposed by the service, is implemented using Vertx [20]. This toolkit has proven properties like multi-language support, flexibility etc., and besides that, it includes an Apache Cassandra Client (CC) [21] in order to facilitate the integration. Moreover, a core component within the toolkit is the Event Bus (EB) that makes the internal message exchange possible in a very resource-efficient and asynchronous way.

The choice of Apache Cassandra (CS) as a NoSql distributed database backend is based on its maturity, high availability, and scalability. The service virtualization as whole is achieved using Docker [22] container Instances (DI) for the lightness and robustness it shows.

The RESTful service EP is used over HTTP/TCP/IPv6/GbE where the interface and the IPv6 addresses are isolated to make the experiment as unaffected by any other traffic as possible. The clients access the operations of the service in a multi-threaded way and the traffic clients create is consisted by create operations, i.e., POST request as the request/response pattern shown in Fig. 7.

The operation, shown in Fig. 7, is consisted by a request, that includes the EP's URI, headers like Host, Content-type, Accept, etc., plus an experimental header to keep track of the moment the request is created (subsequently copied into the response as it is received by the EP) in a local for the respective client nano-time scale, and JSON description of the operation placed in the request body part.

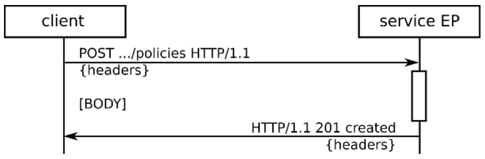

Fig. 7. Typical Policy create operation pattern as message exchange of HTTP POST method

The traffic that is generated for latency estimation purposes is of volume twenty thousand operations of creation type, i.e., couple of request/answer. Thus, the recorded latency for each operation is stored as nanoseconds, where the 4th and 18th time frames of one thousand operation's latencies are shown in Figs 8 and 9, respectively.

The 4th and 18th groups of latency values, formed by one thousand operations each, have been selected because of the following:

a) the initial warm-up of the system (when components that implement lazy instantiation or others with low initial parametric values, like a hash-map constructed with default volume, are not ready yet, and thus it might affect on the measured latency values) normally takes time and so the first three groups are skipped but the fourth is taken as a representative group;

b) having passed the initial transition phase the system goes into a fully-loaded state by the incoming sequences of operations, and hopefully enters its so-called steady state, which has to be well later than the fourth group, so a representative group for this state is chosen to be the eighteenth one.

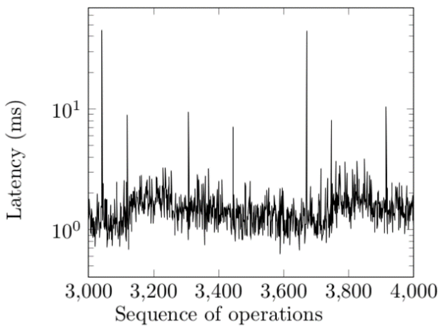

Fig. 8. Sequence of latency values, measured within the 4th group of thousand operations

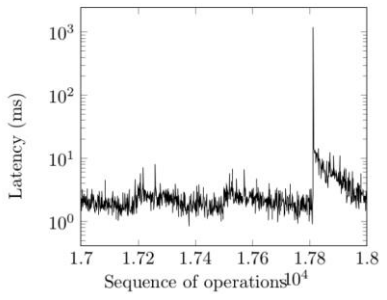

Fig. 9. Sequence of latency values, measured within the 18th group of thousand operations

The sequence of latency values might be seen as a stochastic process, thus a typical tool to depict such a process is to create the respective Probability Density Functions (PDF) as in Fig. 10.

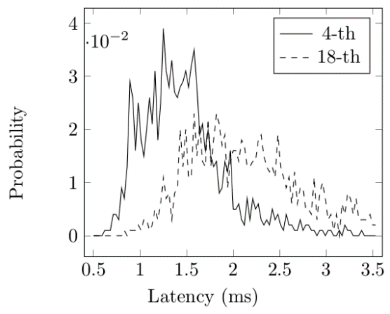

Fig. 10. Probability density functions for latency values, measured in the 4th and 18th groups of thousand operations

Despite of the observable similarity of the curves in their shapes, it is rather to conclude that the process is quite far from the definition for stationary one.

Besides the apparent comparison of PDFs belonging to given test and showing the statistical dynamics of latency values distribution shift toward slightly higher values, the compact representation, even after further data compression by Gaussian Mixture Model, say five component one, might be very much useful when estimating the gain (or loss), in terms of latency, from system reconfiguration or software components' refactoring and upgrade.

Table 1. Average latency values for time frames (in ms)

Group	0-95%	95-100%	0-100%
4th	1.381	3.702	1.613
18th	2.222	19.516	3.951

The latency average values, shown in Table 1, are within the limit of 2-4 ms, when averaged over the whole-time frame, but it is easy to notice that top 5% of latency values give average higher than the lower 95% even in cases up to almost 10 times worse values.

As to [7], control loops exist at various O-RAN levels and run simultaneously. In the context of the use case being considered, they may interact with each other. Currently, there are no feasibility studies to O-RAN implementation, but the O-RAN Use Cases Analysis Report defines relevant interaction for the O-CU-CP and O-DU control loops, responsible for call control and mobility, radio scheduling, etc. along with slower mechanisms involving SMO management interfaces [23]. The timing of the control loops is use case dependent. Typical execution time for use cases involving the non-RT control loops is 1 s or more; near-RT control loops are in the order of 10 ms or more; control loops in the E2 Nodes can operate below 10 ms (e.g., O-DU radio scheduling). So, the measured latency values are acceptable.

6. Conclusion

The O-RAN creates flexible, scalable, and cost-effective network models enabling cost reduction and generation of new service capabilities. The creation of open interfaces breaks the vendor-lock-in of the mobile network operators providing platform for flexibility, interoperability, and agility. The intelligent connectivity enables communications for many critical infrastructures and promotes a new environment for service innovations.

As a new concept, O-RAN architecture can accommodate many new use cases aimed at optimization of Radio Resource Management. The paper presents a use case where future O-RAN implementations may be used for dynamic RRC control. The research ties the O-RAN embedded intelligence with adaptive configuration of inactivity timer based on prediction of device activity. The rationality of the dynamic RRC control is to consider current radio conditions, cell loads, and the durations while the device is in active state and in idle state. Intelligent AI/ML models trained or updated by non-RT services may reason about near future device activity and may be used to generate and send policies for dynamic RRC to the near-RT services. The near-RT services may run the AI/ML models to evaluate the enforced policy efficiency.

Policy lifecycle models are developed, formally described, and theoretically verified. It is proved that the models which must be supported by the non-RT control and the near-RT control functions provide matching views of the policy status.

The paper evaluates by emulation the performance of the RESTful interface between non-RT services and near-RT service in O-RAN architecture. The results show that the injected latency of the policy management interfaces is around 3 ms.

The use case being considered leverages the unique O-RAN architecture benefits and illustrates how existing RRC adaptive algorithms can be used through standardized and open interfaces in a multi-vendor environment and cloudified RAN. The adaptive RRC control is useful in special cases such as unforeseen and emergency situations, where the preconfigured RAN parameters cannot meet the dynamic performance requirements and where AI/ML models can contribute to improvement of radio resource efficiency.

Acknowledgements**:** This research was supported by the Bulgarian National Science Fund under Grant No KP-06-H37/33.

References

1. Nikman, S., et al. Intelligent O-RAN for Beyond 5G and 6G Wireless Networks. – Signal Processing (eess.SP); Machine Learning (cs.LG), 2020, pp.1-7.
2. Abualhaj, M. M., M. M. Al-Tahrawi, A. H. Hussein, S. N. Al-Khatib. Fuzzy-Logic Based Active Queue Management Using Performance Metrics Mapping into Multi-Congestion Indicators. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 2, pp. 29-44.
3. Munusamy, N., S. Vijayan, M. Ezhilarasi. Role of Clustering, Routing Protocols. MAC Protocols and Load Balancing in Wireless Sensor Networks: An Energy-Efficiency Perspective. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 2, pp. 136-165.
4. Park, S.-H., S. Jeong, J. Na, O. Simeone, S. Shamai. Collaborative Cloud and Edge Mobile Computing in C-RAN Systems with Minimal End-to-End Latency. – IEEE Transactions on Signal and Information Processing over Networks, Vol. **7**, 2021, pp. 259-274.
5. Gavrilovska, L., V. Rakovic, D. Denkovski. From Cloud RAN to Open RAN. – Wireless Personal Communications, Vol. **113,** 2020, pp. 1523-1539.
6. Jermin Jeaunita, T. C., V. Sarasvathi. A Multi-Agent Reinforcement Learning-Based Optimized Routing for QoS in IoT. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 4, pp. 45-61.
7. O-RAN.WG1. O-RAN-Architecture-Description-v05.00 Technical Specification. O-RAN Architecture Description, O-RAN Alliance, 2021.
8. O-RAN.WG2. Use-Case-Requirements-v04.00-v02.00 O-RAN Working Group 2 Non-RT RIC A1 Interface: Use Cases and Requirements. Technical Specification. O-RAN Alliance, 2021.
9. 3GPP TS 38.401 Technical Specification Group Radio Access Network; NG-RAN; Architecture Description, Release 16, v16.7.0, 2021.
10. Rinaldi, F., A. Raschellà, S. Pizzi. 5G NR System Design: A Concise Survey of Key Features and Capabilities. – Wireless Networks, Vol. **27**, 2021, pp. 5173-5188.
11. Gures, E., I. Shayea, A. Alhammadi, M. Ergen, H. Mohamad. A Comprehensive Survey on Mobility Management in 5G Heterogeneous Networks: Architectures, Challenges and Solutions. – In: IEEE Access. Vol. **2020**. 2020, pp. 195883-195813.
12. Dahlman, E., S. Parkvall, J. Skold. Radio-Interface Architecture. Edited Book 5G NR. The Next Generation Wireless Access Technology. Second Edition. Academic Press, 2021, pp. 79-113.
13. Khlass, A., D. Laselva. Efficient Handling of Small Data Transmission for RRC Inactive UEs in 5G Networks. – In: Proc. of IEEE 93rd Vehicular Technology Conference (VTC'21-Spring), 2021, pp. 1-7.
14. Ryoo, S., J. Jung, R. Ahn. Energy Efficiency Enhancement with RRC Connection Control for 5G New RAT. – In: Proc. of IEEE Wireless Communications and Networking Conference (WCNC'18), 2018, pp. 1-6.

15. C a o, L., R. L i, J. C r o w c r o f t, Z. Z h a o, H. Z h a n g. Intelligent Slicing of Radio Resource Control Layer for Cellular IoT: Design and Implementation. Cornell University, Information Theory, Networking and Internet Architecture, 2020, pp. 1-14.
16. R a g o, A., P. V e n t r e l l a, G. P i r o, G. B o g g i a, P. D i n i. Towards an Optimal Management of the 5G Cloud-RAN through a Spatio-Temporal Prediction of Users' Dynamics. – In: Proc. of Mediterranean Communication and Computer Networking Conference (MedComNet'20), 2020, pp. 1-4.
17. O-RAN.WG2, A1GAP-v02.03 O-RAN Working Group 2 (Non-RT RIC and A1 Interface WG). A1 Interface: Application Protocol, Technical Specification, O-RAN Alliance, 2021.
18. Z h a n g, K., T. L i u, D. C h e n g. Observability of Finite Labelled Transition Systems. – IEEE Transactions on Automatic Control, Vol. **63**, 2018, No 6, pp. 1591-1602.
19. S c h a f t, A. J. Equivalence of Dynamical Systems by Bisimulation. – IEEE Transactions on Automatic Control, Vol. **49**, 2004, No 12, pp. 2160-2172.
20. Eclipse Foundation. Vertx (Online).
https://vertx.io/
21. Apache Foundation. Cassandra (Online).
https://cassandra.apache.org/
22. Docker Inc. Docker CommunityEdition (Online).
https://www.docker.com/
23. O-RAN.WG1.Use-Case-Analysis-Report-v03.00. Technical Report. O-RAN Working Group 1 Use Case Analysis Report, 2020.

Received: 20.04.2022; Second Version: 29.06.2022; Accepted: 08.07.2022 (fast track)