This is a repository copy of *Towards Achieving Energy Efficiency and Service Availability in O-RAN via Formal Verification*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/224385/

Towards Achieving Energy Efficiency and Service Availability in O-RAN via Formal Verification

Roberto Metere[1], Kangfeng Ye[1], Yue Gu[2], Zhi Zhang[3], Dalal Alrajeh[3] , Michele Sevegnani[2], Poonam Yadav[1]

[1]University of York, UK
[2]University of Glasgow, UK
[3]Imperial College London, UK

Abstract

As Open Radio Access Networks (O-RAN) continue to expand, AI-driven applications (xApps) are increasingly being deployed enhance network management. However, developing xApps without formal verification risks introducing logical inconsistencies, particularly in balancing energy efficiency and service availability. In this paper, we argue that prior to their development, the formal analysis of xApp models should be a critical early step in the O-RAN design process. Using the PRISM model checker, we demonstrate how our results provide realistic insights into the thresholds between energy efficiency and service availability. While our models are simplified, the findings highlight how AI-informed decisions can enable more effective cell-switching policies. We position formal verification as an essential practice for future xApp development, avoiding fallacies in real-world applications and ensuring networks operate efficiently.

Keywords: Formal Verification Probabilistic Model Checking PRISM O-RAN xApp Energy Efficiency

1 Introduction

The evolution of Open Radio Access Networks (O-RAN)[1] is accelerating with deployments by multiple operators, such as Vodafone, in worldwide locations [27, 4, 1], including the UK [34]. O-RAN integrates AI-based applications (xApps) to enhance the responsiveness and efficiency of network management [28]. Although machine learning models have been used to predict the behaviour of 3G, 4G, and 5G networks [32], O-RAN's approach to AI integration in 6G significantly advances real-time network adaptability [10]. We do not explicitly model

---

[1]A table in Section 5 lists all acronyms in the paper.

such AI-based predictors, but model the probabilistic nature of their output, as accompanied by some level of confidence. Precisely, we assume that AI-based xApp can learn probabilities for User Equipment (UEs) being on or off, and such probabilities are used to control the network.

Traditional cellular networks often fail to capture real-world variability [39] and usually use static rule-based policies, such as indefinitely switching off radio cells (RCs) when no users are connected. These static policies cannot analyse dynamic behaviour, resulting in unclear situations for when and how long before reactivating RCs. This leads to issues with network availability, compromises in quality of service (QoS), and inefficiencies in power consumption. For example, highly frequent RC on/off can result in the deterioration of user-perceived delay, unnecessary frequent handovers of UEs, and higher costs associated with RC mode transitions.[29, 15] Achieving a balance between availability, QoS, and power efficiency has become a critical challenge for the communications industry.

There have been significant contributions toward addressing these issues in different contexts [33, 24, 7, 31, 19]. Recent efforts, such as those in [4, 7], have explored how the O-RAN architecture can support power management and how implementing specific architectural approaches can drastically reduce energy consumption while meeting the requirements for QoS and service availability. Additionally, Zhang et al.[18] optimise strategies for switching RCs on and off, as well as user association policies, while ensuring users' QoS is maintained. However, traditional optimisation methods often demand significant computing power and struggle to adapt to the dynamic and evolving nature of network environments.

In contrast, ML-based optimisation methods can overcome such difficulties. In [35], an intelligent network application utilising deep learning enables network slicing in O-RAN, allowing emerging IoT services to coexist while ensuring compliance with required service level agreements. Sbella et al. [30] employ supervised and unsupervised learning technique to predict the network behaviour and dynamically switch base stations on and off to conserve energy in a mobile cellular network. However, this policy may affect the QoS due to the switching off at the macro layer. In the context described above, O-RAN and AI-based management applications introduce new challenges that traditional models did not face. One example is misconfiguration risks [38]. Our work aims to bridge this gap by emphasising the role of formal verification in developing xApps for O-RAN. This links to a lack of policy that accounts for the necessary dynamics of O-RAN networks. The ML is adopted to predict the dynamics of O-RAN networks based on the data set, and then the formal verification generates the decision based on the prediction.

We strongly advocate for formal verification as an essential part of ensuring that AI-based xApps for O-RAN avoid critical logical fallacies during development. By analysing dynamics early, we can prevent future availability issues caused by static cell-switching policies. In this paper, we demonstrate, through simulations with a well-known formal verification tool, the PRISM model checker [20] that AI-based decisions can improve energy efficiency and availability, providing robust results that can guide real-world development.

In this paper, we outline key contributions that support our position:

- We argue that formal verification of xApp models, using tools like PRISM, should be the first step in the development of AI-driven applications for O-RAN.

- We present simulations that highlight how AI-based predictions can enhance cell-switching policies. By predicting user behaviour, RCs can make informed decisions about when to switch on or off, improving both energy efficiency and service availability.

- Our analysis of energy efficiency versus availability thresholds reveals that static policies often lead to self-imposed denial-of-service scenarios. Incorporating AI-based predictions mitigates these risks and ensures better resource management.

The remainder of this paper is organised as follows. Section 2 provides the necessary background about O-RAN, energy management, and probabilistic model checking for further presentation of our work in the subsequent sections. In Section 3, we introduce a dynamic UE scenario, its corresponding management policy, and discuss considerations in modelling. After that, we present the PRISM model in Section 4 and discuss the interesting properties and related verification results. Finally, we discuss future work in Section 5.

2 Background

By integrating the controlling of xApps into O-RAN, one can optimise energy efficiency, as demonstrated by two proposed xApps in [23]: one switches unused RCs to sleep mode, and the other reallocates any UEs while maintaining QoS and balanced RC workload. We first present an overview of the structure and components of O-RAN, to link back to current methods for energy management.

2.1 Overview of O-RAN

O-RAN adopts and supports the 3rd Generation Partnership Project (3GPP) functional split, in which base station (BS) functions are virtualised as network elements and distributed across various network nodes: the central unit (CU), distributed unit (DU), and radio unit (RU) [6, 28], as shown in Figure 1. The CU is typically situated in a central location and handles the higher-layer functions of the RAN. It manages multiple distributed units, enhancing scalability and optimising resource utilisation. By contrast, the DU is responsible for lower-layer functions such as the radio link control, medium access control, and parts of the physical layer, and is closer to the RU. It handles the real-time processing and efficient management of data transmission between the CU and RU. The RU, located at the network's edge, manages the radio frequency processing and transmission over the air interface between the BS and UE. The core innovation of O-RAN is improving the RAN performance by utilising the above-virtualised
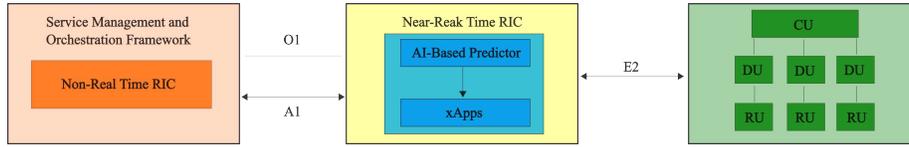
Figure 1: O-RAN Architecture

network elements and open interfaces, integrating intelligence into the RAN through introducing the RAN Intelligent Controller (RIC) [5]. This innovation allows operators to implement and deploy custom RAN functions composed of virtual network functions. Besides that, intelligence is required to handle demanding service requirements [26].

Finally, utilising the RIC in the RAN has the potential to enhance energy efficiency in modern telecommunication systems within the O-RAN architecture. In [23], two xApps are proposed to facilitate RC switching, aiming to reduce power consumption of static model with fixed location of UEs. The first xApp is designed to switch RCs with a lower number of UEs into sleep mode, while the second xApp reallocates the UEs to some RCs, while ensuring that the QoS for each UE is maintained, and the workload for each RC remains within a specific limit.

2.2 Formal Methods and PRISM

In our context, formal methods refer to mathematically rigorous techniques used in network management to ensure system correctness, reliability, and other properties. As modern telecommunications systems, such as O-RAN with applications for dynamic control, become increasingly complex, formal methods are essential to mitigate potential risks like misconfigurations or unintended behaviour.

These methods can be applied to validate protocols, optimise resource allocation, and ensure energy-saving strategies do not compromise service availability. By offering precise mathematical models of system behaviours, formal methods help network engineers analyse edge cases that might be missed by traditional testing techniques. Formal verification techniques, such as model checking, offer an exhaustive approach to system validation in dynamic environments [11].

In this work, we model our scenarios with PRISM [21], a leading tool for probabilistic model checking, used to formally verify systems with random or uncertain behaviours. In network management, where uncertainty prevails (such as fluctuating user demands or varying conditions) probabilistic model checking provides a means to model systems as stochastic processes, e.g. Markov chains, and to enable the verification of performance metrics like reliability, availability, and energy efficiency. In the O-RAN network systems, PRISM is particularly relevant in scenarios where (AI-driven or not) decisions are based on probabilistic models of user behaviour.

2.3   Probabilistic Model Checking

Our models take the form of (labelled) continuous-time Markov chains (CTMCs) to represent the real-time behaviours of users and RCs. A CTMC consists of a finite set of states, with a designated initial state, and a function that assigns a transition *rate* λ between any two states. A transition between states s and s' can occur only if $\lambda(s, s') > 0$, in which case the probability of this transition happening within t time units is $1 - e^{-\lambda(s, s') \cdot t}$. Essentially, this means transitions with a higher rate occur more frequently. We label states with atomic predicates to simplify the representation of logical formulae over the CTMC, e.g. $s \mapsto \{\text{steady}\}$.

Instead of constructing a CTMC directly, we utilise the PRISM modelling language [20], a state-based language based on Reactive Modules [2], that facilitates high-level process specifications. Processes are depicted by modules consisting of non-deterministic choices among action-labelled guarded commands that denote transitions. For example, a module might represent a user with an internal state, e.g., connection, and transitions between internal states at a specified rate modelling the handover. PRISM has been successfully used in a range of domains including real-time probabilistic communication protocols [13], biological systems [16] and human-swarm interactions [14]. It is applicable and expressive enough to perform an exhaustive analysis of all possible behaviours of the system under study [20].

A core component of PRISM is a model checker that allows system behaviours to be quantified [22], e.g., querying the probability a system reaching a steady status. Properties of interest are expressed in an extended form of Continuous Stochastic Logic (CSL) [3], which is a temporal logic with probabilistic operators. For example, the expression $\mathbf{F}\,\varphi$, where $\mathbf{F}$ is temporal operator meaning *eventually*, asserts that, for all paths, we eventually reach a state in which φ is true. Quantitative properties are specified with the operator $\mathcal{P}_{=?}$. For instance $\mathcal{P}_{=?}[\varphi]$ represents an expression on the *likelihood* of a path existing where φ is true, rather than *if* a path exists where φ is true. It also possible to express bounded variants, e.g., the property $\mathcal{P}_{=?}[\mathbf{F}^{\leq t}\,\varphi]$ states the likelihood of φ to be true within t time units. The steady-state behaviour of a model is examined by $\mathcal{S}$ operator, similar to $\mathcal{P}$, which can also be applied to quantify the probability, e.g. property $\mathcal{S}_{=?}[\varphi]$ returns the steady-state probability of being in a state that satisfies φ. PRISM allows *rewards* (*costs*) to be assigned to states. The $\mathcal{R}$ operator can quantify reward-based properties, e.g. $\mathcal{R}_{=?}[\mathbf{S}]$. In addition, we can also customise properties to obtain results for any single state (not only the initial state) of the model. This is achieved by using *filters*, e.g. $filter\,(\,min,\,\mathcal{P}_{=?}[\mathbf{F}^{\leq t}\,\varphi]\,,\,s_1\,)$ gives the minimum value of the probability of reaching φ within t time units from state s_1.

In PRISM, processes are represented by modules consisting of non-deterministic choice over action-labelled guarded commands (which denote transitions); modules are composed of all common actions. For example, a guarded command has the form:

$$[action]\ guard\ \rightarrow rate\ :\ update$$

meaning the process can make a transition to a state described by the *update* at the given *rate* if the *guard* is true. Transitions in different modules can be synchronised with the same action labels. The rate of the synchronisation is then the *product* of individual rates:

$$[action_1] \ guard_1 \ \rightarrow rate_1 \ : \ update_1$$

$$[action_1] \ guard_2 \ \rightarrow rate_2 \ : \ update_2$$

where the transition to $update_1$ and $update_2$ happens when $guard_1$ and $guard_2$ are true with rate $rate_1 \cdot rate_2$.

3 Dynamic Client Management

As discussed in Section 2.1, the two xApps in [23] consider only the static scenario where the states of UEs are also unchanged (always on). One important consideration in their xApp2 is Reference Signals Received Power (RSRP), which is a measurement of the signal strength of a UE received from an RC and represents the quality of connection between the UE and the RC. A UE can be re-assigned to another RC only if the new connection satisfies the RSRP threshold.

In this paper, we consider scenarios where UEs are dynamically switched on and off. For example, in a sensor network, sensors are regularly on or off to monitor particular events and save energy, but when a UE is switched on or off is uncertain. We assume all UEs and RCs's locations are fixed, as illustrated in Figure 2 where three circles denote the possible coverage areas by the three RCs. We further assume the RSRP for a potential connection from each UE to an RC (that covers the UE) satisfies the required minimum threshold so RSRP is not considered in our scenario.

We design a policy to minimise the total power consumption by RCs while maintaining the QoS for each UE. In this policy,

PO1 each RC has a serving capacity; the RC will not accept a new connection from any UE if it is full in capacity.

PO2 an RC will be on standby if all its covered UEs are off.

PO3 when a UE is switched on,

> **PO3.1** it will randomly choose an RC (that covers it, is on, and not full in capacity) to connect.
>
> **PO3.2** if no such RC exists, it will randomly choose an RC (that covers it and is on standby) to connect and at the same time, the RC is switched to on from standby.
>
> **PO3.3** otherwise, the UE fails to connect to any RC so it is out of service.
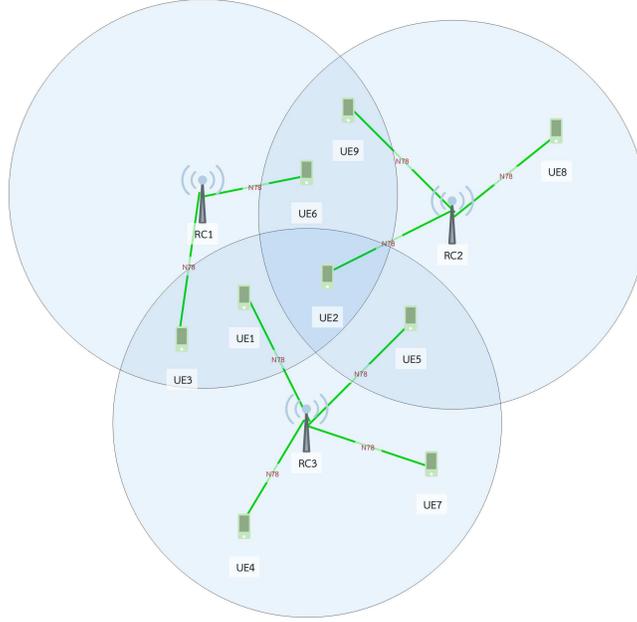
Figure 2: A scenario with 3 RCs and 9 UEs where their locations are all fixed. The diagram shows one possible connection where all the RCs are on.

PO4 when a UE is switched off, its current connection to the RC is also released.

For the scenario considered in Figure 2, we expect, at some point, RC1 to be on standby and the UE3 and UE6 it currently serves would be served by RC2 and RC3 respectively. As a result, the total power consumption is reduced.

3.1   Modelling Considerations

We aim to model the dynamic UE management using CTMCs in PRISM and then analyse energy consumption and service availability using probabilistic model checking. To model the scenario in Figure 2, we consider

C1 An RC can be in two states: standby or operating. It also maintains the number of served UEs to ensure its capacity will not be exceeded (otherwise, the service quality for UEs will be deteriorated).

C2 Whether an UE is covered by an RC is recorded in the model but not the exact location of UEs.

C3 A UE can be in one of two states: on and off. But when it will be switched from on to off, or off to on is uncertain, but the probability of leaving on or off follows exponential distributions. We use r_{on} (or r_{off}) to denote a UE's on rate (or off rate). The mean duration of on and off for a UE is $1/r_{on}$ and $1/r_{off}$.

C4 An UE tries to connect to an RC based on the policy **PO3**. If it succeeds, the connected RC is recorded in the UE and hence is served. At the same time, the number of served UEs in the RC is increased by 1.

C5 The number of served UEs in an RC will be decreased by 1 if a connected UE is switched off.

C6 Time consideration: each command in a CTMC model of PRISM must have an associated rate. For the commands used to switch on or off an UE, its associated rate is r_{on} (or r_{off}). Otherwise, each command has a default rate r which is larger than r_{on} and r_{off}. This means it has less time or duration than that for the UE's on or Off time. In this paper, we choose r to be 1.0, and so r_{on} (or r_{off}) should be larger than it. Another reason to choose 1.0 for r is due to the PRISM's system module which is combined and merged from all the modules by multiplying rates from multiple commands. Simply any rate multiplied by 1 is still itself. Otherwise, the merged rate will be scaled.

4 Formal Model and Analysis

In this section, we explain how we model the problem in PRISM, and how we capture power consumption and availability. We also present and discuss our results. We remark that our quantitative results are not limited to our model and offer important insights for policies that have to balance between power consumption and service availability.

4.1 PRISM Model Description

In PRISM, we model each RC or UE as a module. The behaviour of a RC is defined using a state machine shown in the top diagram in Figure 3 and that of a UE is defined in the bottom diagram. The connections between these modules are illustrated in the middle diagram in Figure 3. A connection between a RC module and a UE module denotes a synchronisation using actions between two modules. There are three possible actions for each connection: such as `ue9_to_rc3` for UE9 to connect to RC3 when RC3 is on, `ue9_on_rc3` for UE9 to connect to RC3 when RC3 is off, and `ue9_from_rc3` for UE9 to disconnect from RC3.

As given in the RC state machine in the top diagram of Figure 3, initially it is `Off` with serving no UE (`S=0`) and can be switched on by a demand from any UE through synchronisation over `ue_on_rc`. At the same time, the UE is connected to the RC and the `S` is increased to 1. Before the RC reaches its capacity (`S=C-1`), it allows UE to disconnect from it (so `S` decreases) and UE to connect to it (so `S` increases). If $S = 0$, then it is switched off.

The state machine of a UE, as illustrated in Figure 3 also has two states: `Off` and On. The switch between them is not triggered by actions but uncertainly follows exponential distributions with a rate `ue_on_rate` for `Off` to `On` and a rate `ue_off_rate` for `O1` to `On`. The UE is initially `Off` and will be switched on
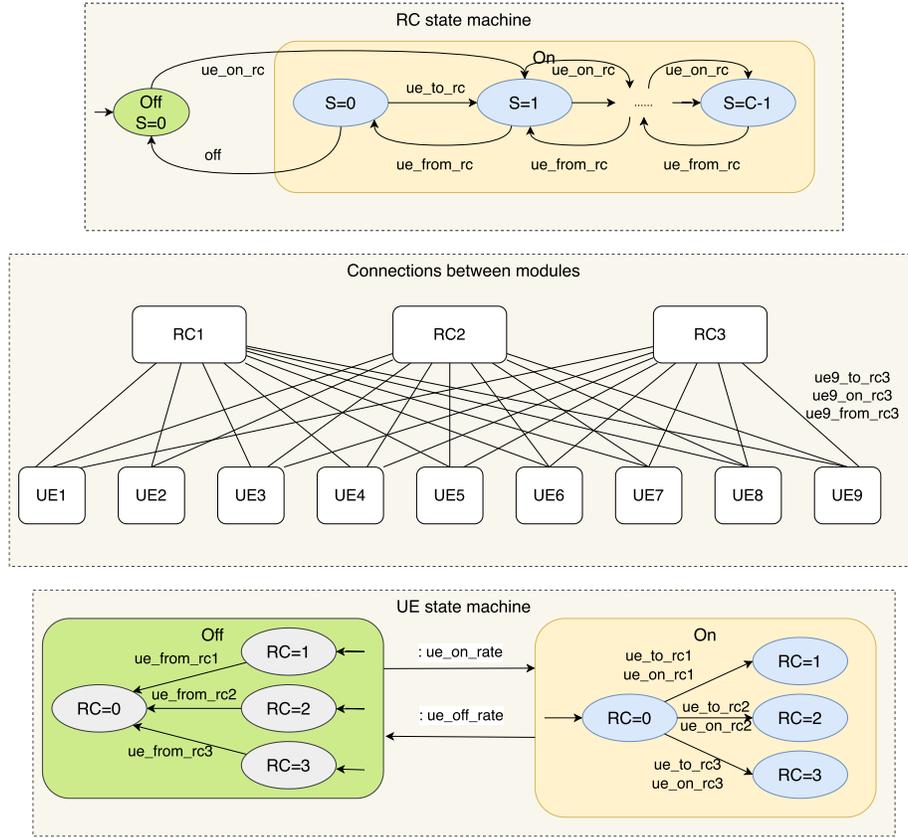
Figure 3: PRISM module connections and behaviours.

following exponential distributions. Each `Off` or `On` contains four substates to record the connected RC id or not connected (`RC=0`).

We show the modelling of an RC in PRISM in Figure 4.[2] In the beginning part on lines `#2-6` of the snippet, we define some constants: the default rate and the number of RCs (set to 1 and 3 respectively on line `#2`), the identity for each RC on line `#3`, the serving capacity for each RC on line `#4`, and the power consumption of an RC in its operating and standby states (on lines `#5` and `#6`).

The module `rc1` on lines `#7-24` contains two local variables: `rc1_on` to record the RC's state and `rc1_serving` to record the number of UEs in serving. The module models the behaviours of RC1. The command on line `#11` turns the RC to standby and clears the serving record (corresponding to policy **PO2**). The commands on lines `#13-15` connect a UE to this RC and increase `rc1_serving` if it is not full in capacity (corresponding to policy **PO3.1**). The

---

[2]The complete model is available on GitHub at `https://github.com/RandallYe/ORAN-xApp-QoS-PRISM-Models/blob/master/DataMod24/ue_dynamics_9_3_cnf1.prism`.

```prism
1  ctmc
2  const double default_rate = 1; const int N_RCs = 3;
3  const int rc1_id = 1; const int rc2_id = 2; const int rc3_id = 3;
4  const int rc1_capacity = 5; const int rc2_capacity = 5; ...
5  const double power_consumption_rc_on = 1;
6  const double power_consumption_rc_off = 0.01;
7  module rc1
8    rc1_on : bool init false; // standby - false, on - true
9    rc1_serving: [0..rc1_capacity] init 0;
10   // rc1 serves no UEs, switch off
11   [rc1_off] (rc1_serving = 0) & rc1_on -> default_rate: (rc1_on' = false) &
       ↪ (rc1_serving' = 0);
12   // A new ue is linked to this rc1
13   [ue1_to_rc1] (rc1_serving < rc1_capacity) & rc1_on -> default_rate:
       ↪ (rc1_serving' = rc1_serving + 1);
14   [ue2_to_rc1] (rc1_serving < rc1_capacity) & rc1_on -> default_rate:
       ↪ (rc1_serving' = rc1_serving + 1);
15   ...
16   // Switch on is on demand
17   [ue1_on_rc1] (rc1_serving = 0) & (rc1_on = false) -> default_rate: (rc1_on' =
       ↪ true) & (rc1_serving' = rc1_serving + 1);
18   [ue2_on_rc1] (rc1_serving = 0) & (rc1_on = false) -> default_rate: (rc1_on' =
       ↪ true) & (rc1_serving' = rc1_serving + 1);
19   ...
20   // A UE is disconnected from this rc1
21   [ue1_from_rc1] (rc1_serving > 0) & rc1_on -> default_rate: (rc1_serving' =
       ↪ rc1_serving - 1);
22   [ue2_from_rc1] (rc1_serving > 0) & rc1_on -> default_rate: (rc1_serving' =
       ↪ rc1_serving - 1);
23   ...
24 endmodule
```

Figure 4: The modelling of an RC in PRISM and the snippet for other two RCs are omitted.

commands on lines #17-19 change the RC to the operating state by a UE and increase `rc1_serving` if the RC is in standby (corresponding to policy **PO3.2**). The commands on lines #21-23 decrease `rc1_serving` if an UE leaves this RC (corresponding to consideration **C5**). We note that each command has an associate label such as `rc1_off` for the command on line #11 and `ue1_to_rc1` for the command on line #13. These labels could be used for synchronisation with other modules such as `ue1_to_rc1` used for synchronisation with a module called `rc1` (whose model will be shown later in Figure 5), or simply to facilitate the definition of transition rewards or simulation.

The modelling of an UE is illustrated in Figure 5 where the constants on lines #1 and #2 denote r_{on} and r_{off} for UE1 (whose values are 0.01 and 0.1, denoting their mean durations are 100 and 10 units of time for the ON and OFF of UE1). The constants on lines #3-5 denote whether UE1 can be served by each RC (the values `true`, `false`, and `true` means UE1 can be served by RC1 and RC3, but not by RC2).

```
1  const double ue1_off_rate = 0.01;
2  const double ue1_on_rate = 0.1;
3  const bool ue1_within_rc1 = true;
4  const bool ue1_within_rc2 = false;
5  const bool ue1_within_rc3 = true;
6  module ue1
7    ue1_on : bool init false; // On and off
8    ue1_rc : [0..N_RCs] init 0; // 0 - Disconnected, or connected RC id
9    [] ue1_on -> ue1_off_rate: (ue1_on' = false); // On -> Off
10   [] (!ue1_on) -> ue1_on_rate: (ue1_on' = true); // Off -> On
11   // Connect to one RC if the RC is on and has available capacity
12   [ue1_to_rc1] (ue1_on) & (ue1_rc = 0) & ue1_within_rc1 & rc1_on & (rc1_serving
     ↪ < rc1_capacity) -> default_rate: (ue1_rc' = rc1_id);
13   ...
14   // Connect if all RCs in the range are off or full capacity, choose one RC
     ↪ that is within the range and off
15   [ue1_on_rc1] (ue1_on) & (ue1_rc = 0) &
16     !(ue1_within_rc1 & rc1_on & (rc1_serving < rc1_capacity)) &
17     !(ue1_within_rc2 & rc2_on & (rc2_serving < rc2_capacity)) &
18     !(ue1_within_rc3 & rc3_on & (rc3_serving < rc3_capacity)) &
19     (ue1_within_rc1 & !rc1_on)
20     -> default_rate: (ue1_rc' = rc1_id);
21   ...
22   // Disconnect from the connected rc
23   [ue1_from_rc1] (!ue1_on) & (ue1_rc = rc1_id) -> default_rate: (ue1_rc' = 0);
24   ...
25 endmodule
```

Figure 5: The modelling of a UE in PRISM and the snippet for other UEs are omitted.

The `ue1` module defined on lines #6-25 contains two local variables: `ue1_on` on line #7 and `uc1_rc` on line #8 to record the current state and the connected RC of UE1. The two commands on lines #9 and #10 switch on and off UE1 with the corresponding rates (see **C3**). The command on line #12 allows UE1 to connect to RC1 (`ue1_rc' = rc1_id`) if the UE is ON (`ue1_on`) and has not been connected (`ue1_rc = 0`), UE1 can be served by RC1 (`ue1_within_rc1`), RC1 is operating (`rc1_on`) and has the available capacity (`rc1_serving < rc1_capacity`) to serve UE1. This command will synchronise (because they have the same label `ue1_to_rc1`) with the command on line #13 in Figure 4 to let UE1 connect to RC1. This command implements policy **PO3.1** and consideration **C4**. There are other two similar commands for UE2 and UE3, omitted here for simplicity. If both RC1 and RC3 are available to serve UE1, which UE1 will be connected is randomly chosen (according to the PRISM's semantics).

The policy **PO3.2** and consideration **C4** is realised through the synchronisation of the command on lines #15-20 with the command on line #17 in Figure 4. This is to deal with a situation when all RCs are either on standby, out of the range, or full in capacity. The command will let UE1 to connect to RC1 if

RC1 is on standby and within the range of UE1. Similarly, there are other two similar commands for UE2 and UE3, omitted here for simplicity. If both RC1 and RC3 are in standby, then which UE1 will be connected is randomly chosen.

The command on line #23 allows UE1 to be disconnected from RC1 if UE1 is currently connected to RC1 and is off now. Similarly, there are other two similar commands for RC2 and RC2.

4.2   Verification and Evaluation

Once the PRISM model is constructed, we can use probabilistic model checking to query interesting qualitative and quantitative properties.

The first property which we are interested in is the average total power consumption of all RCs. We add the following two rewards to the PRISM model which assign `power_consumption_rc_off` and `power_consumption_rc_on` to each RC if it is standby and operating.

```
 1  rewards "standby"
 2    !rc1_on : power_consumption_rc_off;
 3    !rc2_on : power_consumption_rc_off;
 4    !rc3_on : power_consumption_rc_off;
 5  endrewards
 6  rewards "operating"
 7    rc1_on : power_consumption_rc_on;
 8    rc2_on : power_consumption_rc_on;
 9    rc3_on : power_consumption_rc_on;
10  endrewards
```

Table 1: Interested properties and their CSL formulas.

Property	CSL formula
P1	R{"standby"}=?[S]
P2	R{"operating"}=?[S]
P3.1	S=? [rc1_on]
P3.2	S=? [rc2_on]
P3.3	S=? [rc3_on]
P4.1	filter(min, P=? [F<=t ue1_rc>0], ue1_on&ue1_rc=0)
P4.2	filter(min, P=? [F<=t ue2_rc>0], ue2_on&ue2_rc=0)
...	...
P4.9	filter(min, P=? [F<=t ue9_rc>0], ue9_on&ue9_rc=0)

Then we can use CSL formulas R{"standby"}=?[S] or R{"operating"}=?[S] to get the long-run average power consumption per one unit of time, as shown in Table 1 and denoted as P1 and P2. The total power consumption is just their sum: P1 + P2. P1 (or P2) provides us with the total power consumption for all three RCs if they are on standby (or operating). We may also want to know the percentage each RC contributes to P1 and P2. This can be queried using the formula S=? [rc1_on] which means the long-run probability of being in a

state when RC1 is operating. Similarly, we can query that for RC2 and RC3. We use P3.1, P3.2, and P3.3 to denote the property for each RC.

In addition to power consumption, the service availability for each UE is also an important measurement for a policy. We expect every UE can be connected to an RC within an amount of time. A formula `filter(min, P=? [F<=t ue1_rc>0], ue1_on & ue1_rc=0)` gives the `minimum` probability (`P=?`) of UE1 finally being connected to any RC (`ue1_rc>0`) within t units of time (`F<=t`) when it is switched on but not connected (`ue1_on&ue1_rc=0`)). This represents the lower-bound guarantee of the service. A `filter` in PRISM allows us to quantify properties from any state (not only from the initial states of the model usually). In P4.1, the time 0 starts from the states in which UE1 is on but not yet connected. P4.1, P4.2, or . . . P4.9 are properties for different UEs in a same configuration.

Configurations. We consider six configurations of constants shown in Table 2. In this, Cnf1 is a base configuration and other configurations have only one change based on Cnf1. Cnf2 changes the RC capacity from 5 to 6. Cnf3 changes the RC capacity from 5 to 4. Cnf4 increases the rate r_{off} from 0.1 to 1 (so UEs are mostly ON and will only be switched off very quickly). Cnf5 swaps the ON and OFF rates. Cnf6 considers UE3 only being in RC1.

Table 2: Constant values for six configurations where r is for the default rate, Power on and off for `power_consumption_rc_on` and `power_consumption_rc_off`, and Location for the relative position of UEs.

Id	r	Capacity	Power on	Power off	r_{on}	r_{off}	Location
Cnf1	1	5	1	0.01	0.01	0.1	as Figure 2
Cnf2	1	6	1	0.01	0.01	0.1	as Figure 2
Cnf3	1	4	1	0.01	0.01	0.1	as Figure 2
Cnf4	1	5	1	0.01	0.01	1	as Figure 2
Cnf5	1	5	1	0.01	0.1	0.01	as Figure 2
Cnf6	1	5	1	0.01	0.01	0.1	UE3 to be only in RC1

Power consumption. Model checking shows that the value (0.007346391) of P1 is negligible compared to that (2.265360891) of P2 in Cnf1. So here we only discuss P2 and P3. Figure 6 shows the analysis results of P3 for each RC in the six configurations. The sum of three sections for P3.1, P3.2, and P3.3 is the total height of the bar for a configuration and is also the total power consumption (P2) of RCs because `power_consumption_rc_on` is set to 1 in all configurations. From the diagram, we conclude that Cnf6 has the highest energy consumption, followed by Cnf3. We also note that Cnf5 uses the least energy. This is because Cnf6 has UE3, UE4, UE7, and UE8 only within the range of RC1, RC3, and RC2. Therefore, all the three RCs are operating nearly all the time. Cnf3 has the lowest capacity (4) for each RC. To serve all UEs, similarly, all the RCs are

operating most of the time. Cnf5 has UEs mostly OFF (its mean duration is 100) compared to the mean duration (10) of ON. So Cnf5 has the lowest power consumption because RCs can be in standby when all UEs are off. We also observe that the results of P3.2 and P3.3 are always very close, but that of P3.1 has changed dramatically. This is because both RC2 and RC3 in Figure 2 have at least one UE (UE8, or UE4 and UE7) that is within their ranges but cannot be reached by other RCs. In Cnf2, P3.1 is very small and so RC1 is almost not operating, thanks to the high capacity (6) in Cnf2 (so RC2 and RC3 always can serve all UEs).
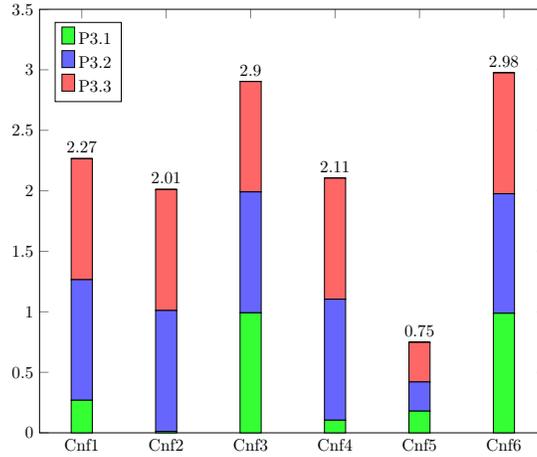
Figure 6: Comparison of power consumption of each RC for six configurations.

Service availability. We show the results of P4 for each UE in Figure 7 when using Cnf1. For UE4 and UE7, the minimum probability is increased linearly in terms of time (t). It can reach nearly 0.5 after 20 units of time. Their probabilities are relatively small compared to that of other UEs. For other UEs, their minimum probability rises sharply and can reach 0.97 after 4 units of time. This reflects UE4 and UE7 are not well served (because they can only be served by RC3) but others are. This is due to the capacity of RC3 is 5 while it covers 6 UEs. At some points, UE4 or UE7 cannot be served by RC3 because it might be full in capacity when UE4 or UE7 tries to connect.

In Figure 8, we show the results of P4 for UE4 (which is only within the range of RC3) in six different configurations. In Cnf2 and Cnf6 (the tallest two overlapped lines), UE4 is well served and highly likely it can connect to a RC within 4 units of time because RC2 has enough capability to allow all UEs within its range (including `ue4`) to join the network. P4 of UE4 for other configurations are less well served. P4 in Cnf3 and Cnf4 has the least minimum probabilities because RCs have a limited capacity 4 in Cnf3 and UEs are mostly on in Cnf4.
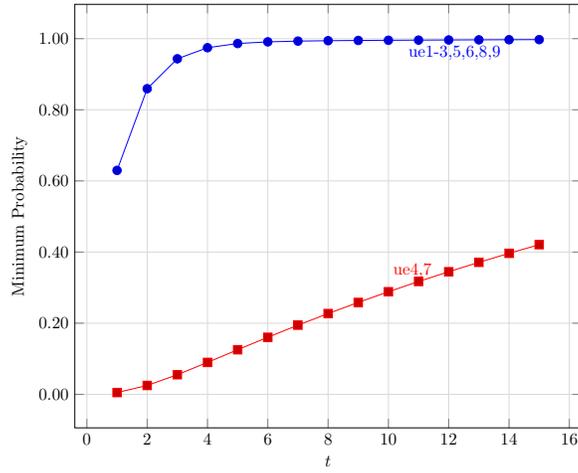
Figure 7: Comparison of P4 for each UE in Cnf1 where the lower line corresponds to (overlapped) UE4 and UE7 and the upper line corresponds to other (overlapped) UEs.
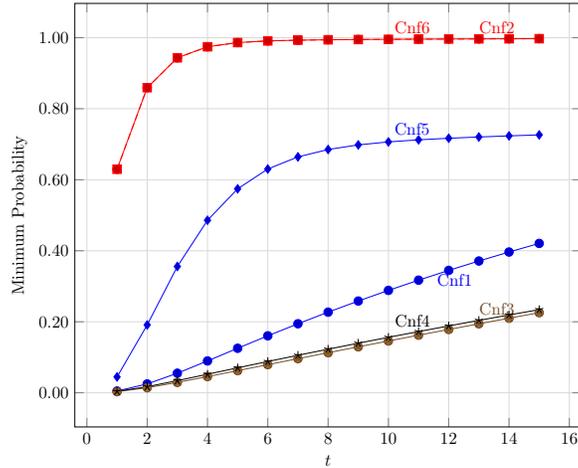
Figure 8: Comparison of P4 for UE4 in different configurations.

Similarly, the P4 for UE8 (which is only within the range of RC2) in different configurations is shown in Figure 9. The UE8 is well served in Cnf1, Cnf2, Cnf4, and Cnf6. It is not well connected in Cnf3 because of the lower capacity (4).
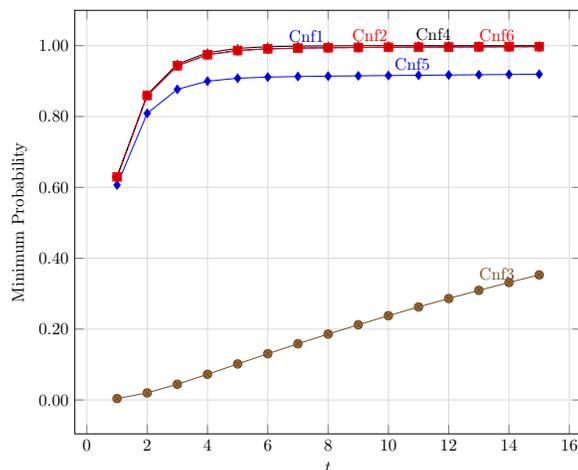
Figure 9: Comparison of P4 for UE8 in different configurations.

4.3 Discussions

From the analysis of results in terms of power consumption and service availability, we conclude that the service capacity of each RC is a very important parameter to consider. Its capacity should be larger than the number of all UEs within its range. The location of UEs and RCs is also equally important. If each UE can be served by more than one RC, then some RCs could be in standby to save energy.

It is noteworthy that our preliminary experiments yield quantitative results in the long term (steady-state analysis) which can be leveraged to strike a balance between quality of service and power consumption. In themselves, these small-scale models do not achieve optimal solutions (not suitable for use as controlling xApps in O-RAN). To arrive to optimal solutions, one must first consider the specific application context, taking into account factors such as priority assignment and minimum bounds for both quality of service and power consumption. Additionally, other relevant metrics should be incorporated to capture the nuances of the scenario under investigation. At present, our models are only capable of providing a minimum guarantee for the network to ensure either service level is met.

In terms of scalability, our experiments showed that our models can suffer the state explosion when the number of users and RCs increases, resulting in a large computation effort. In this case, exhaustive model checking can be costly. To address this problem, PRISM's built-in discrete-event simulator enables the use of *statistical model checking* (SMC) [36], providing approximately accurate results. It effectively samples the model space through repeated simulation instead of exhaustive search [9, 17]. Previous studies have shown that SMC can achieve high performance with large-scale models without a huge cost of

accuracy [14], which is important for practical implementations, considering the requirements of fast response time in O-RAN. However, SMC naturally is not suitable for long-term verification, such as the steady-state analysis, in which case other techniques can be introduced to tackle the state explosion, such as the bonded model checking [12].

5   Conclusion

This work introduces probabilistic model checking, prior to the development of xApps in O-RAN to quantitatively evaluate QoS like energy efficiency and service availability in a given policy. We have based our work on the two xApps discussed in [23] where only static scenarios are considered. In this work, we introduced a dynamic scenario where UEs are regularly (but uncertainly) switched on and off. This requires a new policy to achieve energy efficiency while maintaining service availability. We modelled the scenario and the policy in PRISM and used probabilistic model checking to explore the QoS in terms of different rates, capacities, and locations of UEs and RCs. The analysis results have shown that RC capacity and redundancy are very important design parameters.

As previously discussed, RSRP is an important measurement to maintain service quality for UEs. We have omitted it and assume UEs are static in this work. We will consider a new dynamic scenario where UEs are moving and RSRP is changing too. Though it is not possible to model the exact location of UEs and its dynamics in PRISM, we could model discretised space and verify the impact of a UE's movement on whether the UE's RSRP becomes over and below a threshold. PRISM has been applied to analyse the movement of a robot in two dimensions [37], and Signal Spatio-Temporal Logic (SSTL) [25] can be used to specify spatio-temporal properties for models with linear time and discrete space and verified using SMC.

Our dynamic UE management policy and the analysis results could be helpful to implement a similar scenario in O-RAN simulation test-bed and to evaluate the effectiveness of our work in the real O-RAN network.

While O-RAN xApps are evolving to use machine learning for dealing with the complexity of the network environment. To this end, we will also explore the development of methods for learning and adapting policies at run-time and in response to changes in the network environment, e.g., as in [8], to avoid fallacies and ensure efficient network operation in O-RAN. Future research directions may involve obtaining formally verified outcomes for simplified network architectures, which can then be used to continuously train intelligent xApps and enhance their predictive capabilities.

Acronyms

3GPP	the 3$^{\text{rd}}$ Generation Partnership Project
AI	Artificial Intelligence
BS	Base Station
CSL	Continuous Stochastic Logic
CTMC	Continuous-Time Markov Chain
CU	Central Unit
DU	Distributed Unit
IoT	Internet of Things
O-RAN	Open Radio Access Network
QoS	Quality of Service
RSRP	Reference Signals Received Power
RU	Radio Unit
RC	Radio Cell
RIC	RAN Intelligent Controller
SMC	Statistical Model Checking
UE	User Equipment

Acknowledgments

References

[1] Open Research Area Network (O-RAN) - Map, https://map.o-ran.org/

[2] Alur, R., Henzinger, T.A.: Reactive modules. Formal methods in system design **15**(1), 7–48 (1999)

[3] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on software engineering **29**(6), 524–541 (2003)

[4] Baldini, G., Bolla, R., Bruschi, R., Carrega, A., Davoli, F., Lombardo, C., Rabbani, R.: Toward Sustainable O-RAN Deployment: An In-Depth Analysis of Power Consumption. IEEE Transactions on Green Communications and Networking (2024)

[5] Bonati, L., D'Oro, S., Polese, M., Basagni, S., Melodia, T.: Intelligence and learning in o-ran for data-driven nextg cellular networks. IEEE Communications Magazine **59**(10), 21–27 (2021)

[6] Bonati, L., Polese, M., D'Oro, S., Basagni, S., Melodia, T.: Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead. Computer Networks **182**, 107516 (2020)

[7] Brito, J.A., Moreno, J.I., Contreras, L.M., Caamaño, M.B.: Architecture and Methodology for Green MEC Services Using Programmable Data Planes in 5G and Beyond Networks. In: 2024 IFIP Networking Conference (IFIP Networking). pp. 738–743. IEEE (2024)

[8] Buckworth, T., Alrajeh, D., Kramer, J., Uchitel, S.: Adapting specifications for reactive controllers. In: 18th IEEE/ACM Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2023, Melbourne, Australia, May 15-16, 2023. pp. 1–12. IEEE (2023)

[9] Butkova, Y., Hartmanns, A., Hermanns, H.: A Modest approach to modelling and checking Markov automata. In: International Conference on Quantitative Evaluation of Systems. pp. 52–69. Springer (2019)

[10] Chataut, R., Nankya, M., Akl, R.: 6G networks and the AI revolution—Exploring technologies, applications, and emerging challenges. Sensors **24**(6), 1888 (2024)

[11] Clarke, E.M.: Model checking. In: Foundations of Software Technology and Theoretical Computer Science: 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17. pp. 54–56. Springer (1997)

[12] Clarke, E.M., Klieber, W., Nováček, M., Zuliani, P.: Model checking and the state explosion problem. In: LASER Summer School on Software Engineering, pp. 1–30. Springer (2011)

[13] Duflot, M., Fribourg, L., Herault, T., Lassaigne, R., Magniette, F., Messika, S., Peyronnet, S., Picaronny, C.: Probabilistic model checking of the csma/cd protocol using prism and apmc. Electronic Notes in Theoretical Computer Science **128**(6), 195–214 (2005)

[14] Gu, Y., Hunt, W., Archibald, B., Xu, M., Sevegnani, M., Soorati, M.D.: Successful swarms: Operator situational awareness with modelling and verification at runtime. In: 2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). pp. 541–548. IEEE (2023)

[15] Guo, X., Niu, Z., Zhou, S., Kumar, P.: Delay-constrained energy-optimal base station sleeping control. IEEE Journal on Selected Areas in Communications **34**(5), 1073–1085 (2016)

[16] Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science **391**(3), 239–257 (2008)

[17] Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker storm. International Journal on Software Tools for Technology Transfer pp. 1–22 (2021)

[18] Jian, Z., Muqing, W., Min, Z.: Energy-efficient switching on/off strategies analysis for dense cellular networks with partial conventional base-stations. IEEE Access **8**, 9133–9145 (2019)

[19] Keshta, I., Soni, M., Deb, N., Saravanan, K., Khan, I.R., et al.: Game theory-based optimization for efficient IoT task offloading in 6G network base stations. Measurement: Sensors **33**, 101184 (2024)

[20] Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. pp. 200–204. Springer (2002)

[21] Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23. pp. 585–591. Springer (2011)

[22] Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: International conference on runtime verification. pp. 122–135. Springer (2010)

[23] Liang, X., Al-Tahmeesschi, A., Wang, Q., Chetty, S., Sun, C., Ahmadi, H.: Enhancing Energy Efficiency in O-RAN Through Intelligent xApps Deployment. arXiv preprint arXiv:2405.10116 (2024)

[24] Mohan, P., Rajasekaran, V.A., Santhanam, P., Raja, K.T., Jayagopal, P., Kumar, S., Mallik, S., Qin, H.: TPEMLB: A novel two-phase energy minimized load balancing scheme for WSN data collection with successive convex approximation using mobile sink. Ain Shams Engineering Journal p. 102849 (2024)

[25] Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL. Logical Methods in Computer Science **Volume 14, Issue 4** (Oct 2018). https://doi.org/10.23638/LMCS-14(4:2)2018, https://lmcs.episciences.org/4913

[26] Niknam, S., Roy, A., Dhillon, H.S., Singh, S., Banerji, R., Reed, J.H., Saxena, N., Yoon, S.: Intelligent o-ran for beyond 5g and 6g wireless networks. In: 2022 IEEE Globecom Workshops (GC Wkshps). pp. 215–220. IEEE (2022)

[27] Office, T.P.: TIM: among the first operators in Europe and the only one in Italy to launch Open RAN solutions on the mobile network (2021), https://www.vodafone.co.uk/newscentre/press-release/openran-deployed-in-urban-locations-in-european-first/

[28] Polese, M., Bonati, L., D'oro, S., Basagni, S., Melodia, T.: Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. IEEE Communications Surveys & Tutorials **25**(2), 1376–1411 (2023)

[29] Rached, N.B., Ghazzai, H., Kadri, A., Alouini, M.S.: A time-varied probabilistic on/off switching algorithm for cellular networks. IEEE Communications Letters **22**(3), 634–637 (2018)

[30] Sesto-Castilla, D., Garcia-Villegas, E., Lyberopoulos, G., Theodoropoulou, E.: Use of machine learning for energy efficiency in present and future mobile networks. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC). pp. 1–6. IEEE (2019)

[31] Singh, S.P., Kumar, N., Singh, A., Singh, K.K., Askar, S.S., Abouhawwash, M.: Energy Efficient Hybrid Evolutionary Algorithm for Internet of Everything (IoE)-enabled 6G. IEEE Access (2024)

[32] Tahat, A., Wahhab, F., Edwan, T.A.: An Exemplification of Decisions of Machine Learning Classifiers to Predict Handover in a 5G/4G/3G Cellular Communications Network. In: 2024 20th International Conference on the Design of Reliable Communication Networks (DRCN). pp. 1–5. IEEE (2024)

[33] Thantharate, P., Thantharate, A., Kulkarni, A.: GREENSKY: A fair energy-aware optimization model for UAVs in next-generation wireless networks. Green Energy and Intelligent Transportation **3**(1), 100130 (2024)

[34] Vodafone: Vodafone deploys OpenRAN in urban locations in European first (2022), https://www.vodafone.co.uk/newscentre/press-release/openran-deployed-in-urban-locations-in-european-first/

[35] Yeh, S.p., Bhattacharya, S., Sharma, R., Moustafa, H.: Deep learning for intelligent and automated network slicing in 5g open ran (oran) deployment. IEEE Open Journal of the Communications Society (2023)

[36] Younes, H.L., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: International Conference on Computer Aided Verification. pp. 223–235. Springer (2002)

[37] Younes, H.L.S., Kwiatkowska, M., Norman, G., Parker, D.: Numerical vs. statistical probabilistic model checking **8**(3), 216–228. https://doi.org/10.1007/s10009-005-0187-8

[38] Yungaicela-Naula, N.M., Sharma, V., Scott-Hayward, S.: Misconfiguration in O-RAN: Analysis of the impact of AI/ML. Computer Networks p. 110455 (2024)

[39] Zaki, Y., Pötsch, T., Chen, J., Subramanian, L., Görg, C.: Adaptive congestion control for unpredictable cellular networks. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. pp. 509–522 (2015)