ns3-oran-customizable-db: Uma Ferramenta para Construção de Base de Dados Personalizável para Simulações Open RAN

Felipe G. Táparo[1], **Igor M. Moraes**[2] **e Miguel E. M. Campista**[1]

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
[2]Laboratório MídiaCom
Universidade Federal Fluminense (UFF)

{felipe.taparo,miguel}@gta.ufrj.br, igor@ic.uff.br

Abstract. *An essential part of developing applications for the control and diagnosis of a Radio Access Network (RAN) is simulation. It is crucial to evaluate proposed solutions in a simulated scenario that closely resembles reality to avoid potential failures in implementation. The objective of this work is to add functionalities to the "ns3-oran"extension of the "ns-3"network simulator. The implemented functionalities give users the freedom to use the extension without restrictions on what data can be sent to the Near-RT RIC. The ability to collect, store, and analyze various types of data is fundamental in the development and implementation of machine learning-based solutions. Thus, the developed tools enable the creation of simulation scenarios capable of training reinforcement learning models, using performance metrics that were previously impossible to manipulate within the simulator.*

Resumo. *Uma parte essencial no desenvolvimento de aplicações para o controle e diagnóstico de uma Rede de Acesso via Rádio (Radio Access Network – RAN) é a simulação. É imprescindível a avaliação das soluções propostas em um cenário simulado próximo à realidade a fim de evitar possíveis falhas na implementação. O objetivo deste trabalho é adicionar funcionalidades à extensão "ns3-oran" do simulador de redes "ns-3". As funcionalidades implementadas dão ao usuário a liberdade para utilizar a extensão sem restrições de quais dados podem ser enviados ao Near-RT RIC. A capacidade de coleta, armazenamento e análise de diversos tipos de dados é fundamental no desenvolvimento e implementação de soluções baseadas em aprendizado de máquina. Assim, as ferramentas desenvolvidas permitem a criação de cenários de simulações capazes de treinar modelos por aprendizado por reforço, utilizando métricas de desempenho que antes eram impossíveis de serem manipuladas pelo simulador.*

1. Introdução

A O-RAN é uma arquitetura de rede de acesso via rádio aberta (*Open Radio Access Network* - Open RAN), padronizada pela O-RAN Alliance [O-RAN Alliance 2025],

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FAPESP (2023/00811-0 e 2023/00673-7) e Fundação de Desenvolvimento da Pesquisa - Fundep - Rota 2030.

que define interfaces abertas e, consequentemente, é interoperável e agnóstica a fabricantes de hardware e software. Uma das principais inovações da arquitetura O-RAN é a introdução de dois controladores inteligentes para o gerenciamento da RAN. O controlador inteligente em não tempo-real (*Non Real-Time RAN Intelligent Controller – Non-RT RIC*) e o controlador inteligente em quase tempo-real (*Near Real-Time RAN Intelligent Controller – Near-RT RIC*), que são capazes de tomar ações de controle a partir de aplicações de lógica personalizada, chamadas de rApps e xApps. O Non-RT RIC atua em laços de controle acima de 1 s e hospeda rApps. Já o Near-RT RIC atua em laços de controle entre 10 ms e 1 s e hospeda xApps [Polese et al. 2023]. A arquitetura O-RAN prevê a implementação das rApps e xApps de uma maneira modular, podendo ser implementadas por terceiros a partir de imagens em contêineres, com a característica *open source*. Essas aplicações são utilizadas para controle e diagnóstico da rede. Como exemplo de aplicações que podem ser implementadas em xApps, tem-se o gerenciamento de *handover* e de *beamforming*. A arquitetura O-RAN prevê também a utilização de aplicações de aprendizado de máquina, implementadas como rApps e xApps nos controladores inteligentes para o controle e gerenciamento da RAN.

O processo de avaliação das rApps e xApps é uma etapa essencial no desenvolvimento dessas aplicações. Tal processo pode exigir que as rApps e xApps sejam simuladas em cenários que possam garantir o correto funcionamento na prática. Ademais, aplicações de aprendizado de máquina precisam de um ambiente de testes e, no caso das aplicações que usam o aprendizado por reforço, um ambiente de testes e de treino. Para isso, as simulações proveem um cenário controlado e personalizável, sendo adequado para os testes e o treinamento de aplicações de aprendizado de máquina. Há um problema, porém, no contexto O-RAN para a realização de simulações devido à escassez de simuladores e às limitações dos existentes, como é o caso do ns-3 [ns-3 2025] e da sua extensão ns3-oran [ns3-oran 2025]. O sistema de relatórios e de banco de dados implementados no ns3-oran são restritivos com relação à adição e à recuperação de dados, sendo limitados aos relatórios de (i) localização, (ii) perda de pacotes e (iii) informações de registro de equipamentos de usuários (*User Equipment – UE*) em *evolved Node B – eNBs*, implementados por padrão. Dessa forma, todos os métodos para armazenamento dos relatórios e recuperação de informações são específicos aos já implementados, não sendo genéricos o suficiente para serem usados para novos modelos de relatórios. Essa limitação impede a criação de cenários gerais de simulação e de aplicações, como, por exemplo, rApps ou xApps baseados em aprendizado de máquina, que dependem de outros tipos de informações além dos três tipos previstos pelo simulador.

Este trabalho propõe uma ferramenta para simulação de rede O-RAN baseada na adição de novas funções ao ns3-oran. A ideia base é possibilitar o uso de relatórios de desempenho personalizados. Para isso, a ferramenta proposta adiciona novas funções e estende funções já existentes que visam permitir maior diversidade de tipos de dados na utilização do sistema de banco de dados e de relatórios. Todas as modificações são retrocompatíveis com o ns-o-ran padrão, de modo que os *scripts* de simulação criados utilizando a versão original são compatíveis com a ferramenta de simulação implementada. A fim de demonstrar o funcionamento e a utilidade da ferramenta implementada, dois exemplos de uso são disponibilizados, um evidenciando a utilização de relatórios contendo métricas arbitrárias, e o outro mostrando como as ferramentas implementadas podem ser utilizadas no treinamento de um xApp com aprendizado por reforço.

Este trabalho está organizado da seguinte forma: a Seção 2 explica funcionamento do ns-3 e do ns3-oran. A Seção 3 introduz a arquitetura da ferramenta proposta, enquanto a Seção 4 apresenta os casos de uso e respectivos resultados. Por fim, a Seção 5 conclui este trabalho e esboça possíveis trabalhos futuros. Todo o código desenvolvido está disponível em `https://github.com/felip-T/ns3-oran`. Uma documentação *Doxygen* em HTML com descrições estruturadas de classes e métodos, e uma documentação *Sphinx* com detalhes sobre a implementação da ferramenta e sobre o funcionamento dos exemplos estão disponíveis em `https://github.com/felip-T/ns3-oran/tree/master/doc`. Os exemplos de caso de uso citados neste trabalho estão disponíveis em `https://github.com/felip-T/ns3-oran/tree/master/examples/examples-adaptative`. No repositório também está disponível um Dockerfile que instala o ns-3, a ferramenta e todas as dependências necessárias para a execução dos exemplos.

2. O Simulador ns-3 e a sua extensão ns3-oran

O ns-3 [ns-3 2025] é um simulador de eventos discretos para comunicações em redes. As simulações no ns-3 são realizadas por meio de *scripts* em C++, sendo que a compilação é uma das tarefas do próprio simulador. O ns-3 é um simulador de código aberto e extensível, com diversas extensões disponíveis que adicionam funcionalidades ao simulador. O ns3-oran [ns3-oran 2025] é uma extensão do ns-3 desenvolvida pelo órgão americano NIST (*National Institute of Standards and Technology*), que implementa funcionalidades da arquitetura O-RAN no ns-3 relacionadas ao Near-RT RIC. A implementação do Near-RT RIC do ns3-oran é capaz de trocar informações com os equipamentos de usuário através da interface E2. Vale mencionar que tanto os equipamentos de usuário quanto a interface E2 são também simulados, sendo esta última responsável por interconectar os nós E2 ao Near-RT RIC. O nome de nó E2 é dado a qualquer equipamento conectado à RAN por meio de uma interface E2. A partir da interface E2, o Near-RT RIC recebe relatórios de desempenho dos nós E2 conectados e pode enviar ações de controle aos nós conectados com base na análise dos relatórios recebidos.

O ns3-oran permite a criação de um Near-RT RIC simulado, que se conecta a outros componentes da mesma simulação a partir de uma implementação da interface E2. O ns3-oran implementa também um sistema de relatórios e banco de dados a fim de receber e armazenar relatórios de desempenho dos nós simulados, além de um sistema para implementação de xApps, chamados de *logic modules* pelo simulador.

3. A Ferramenta Proposta

A ferramenta proposta cria dois novos componentes principais, o `OranAdaptativeSqlite` e o `OranReportSqlite`. Juntos, esses dois componentes permitem a criação de relatórios personalizados e a recuperação de informações do banco de dados pelo Near-RT RIC. Vale ressaltar que diversos outros componentes do simulador precisam ser alterados para a adição desse novo sistema de banco de dados, dentre eles, os principais foram a interface E2 e a classe base para banco de dados.

O `OranAdaptativeSqlite` é o novo componente de banco de dados implementado. Esse componente consiste em um banco de dados SQLite que é capaz de gerar tabelas com base em relatórios recebidos em tempo de simulação. Originalmente, todas

as tabelas são criadas no início da simulação e correspondem apenas às informações previstas nos relatórios implementados por padrão no ns3-oran. Com os novos componentes, além das tabelas de relatórios padrão criadas no início da simulação, o banco de dados também é capaz de criar novas tabelas automaticamente, durante a simulação, com base em metadados provindos de relatórios personalizados. A Figura 1 exibe o diagrama UML da classe `OranAdaptativeSqlite`.

A classe `OranAdaptativeSqlite` adiciona métodos para a recuperação de informações no banco de dados. Os métodos iniciados por "+" na Figura 1 são públicos e implementam interfaces para o acesso ao banco de dados. O método `GetLastReport`, recebe uma *string* contendo o nome da tabela ou uma *string* e um número, contendo o nome da tabela e o ID de um nó. O primeiro método retorna um mapa contendo as informações do último relatório inserido na tabela buscada, na qual as chaves correspondem às colunas da tabela SQLite e os valores correspondem às informações contidas nas colunas. Já o segundo método, diferente do primeiro, retorna as últimas informações enviadas por um determinado nó. O método `GetSecondLastReport` é similar ao anterior, recebendo o nome de uma tabela e um ID de um nó. Essa função é útil para comparar variações entre dois relatórios de um mesmo tipo enviados pelo mesmo nó. Por fim, a função `GetCustomQuery` recebe uma *query* de busca arbitrária SQLite e retorna as informações obtidas pela consulta.

Os métodos iniciados por um "#" na Figura 1 são *protected* e são relacionados ao gerenciamento interno no banco de dados. O método `ParseReportTableInfo` é utilizado na criação de novas tabelas no banco de dados, esse método é chamado quando um relatório personalizado nunca antes recebido chega ao Near-RT RIC, e é responsável por processar as informações sobre o tipo de tabela a ser criado para o novo relatório e gerar a *query* para sua criação. O método `CreateReportTable` envia a *query* gerada pelo método `ParseReportTableInfo` para o banco de dados, criando a tabela relacionada ao novo relatório personalizado. O método `ParseReport` é chamado sempre que um relatório chega ao Near-RT RIC e é responsável por processar o conteúdo de um relatório e estruturá-lo de forma a adicionar as informações no banco de dados. O método `CreateReportSaveQuery` utiliza o método `ParseReport` para a criação da *string* de *query* para a adição de informações em uma tabela na chegada de um relatório. Por fim, o método `CreateReportSave` envia a *query* criada pelo método `CreateReportSaveQuery` para o banco de dados e confere se as informações foram adicionadas com sucesso.

O componente `OranReportSqlite` é implementado como uma classe virtual, com o objetivo de ser expandida pelo usuário para a criação de relatórios personalizados. O usuário deve, então, sobrecarregar os métodos `GetTableName`, `GetTableInfo` e `ToString` para a criação de relatórios personalizados. O método `GetTableName` deve ser sobrecarregado de forma que retorne uma *string* contendo o nome da tabela a ser utilizada pelo `OranAdaptativeSqlite` para armazenar os dados vindos desse relatório. O método `GetTableInfo` deve retornar um vetor de tuplas contendo, primeiro, o nome das colunas da tabela onde os dados serão armazenados e, segundo, o tipo do dado a ser armazenado (CHAR, INT, BOOL, ...). Por fim, o método `ToString` deve retornar as informações a serem armazenadas pelo relatório de forma estruturada. Maiores informações sobre a implementação e utilização desses métodos podem ser encontradas

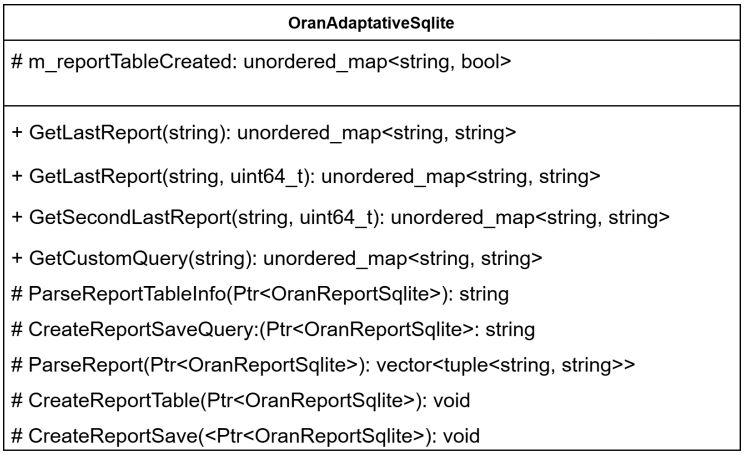

Figura 1. O diagrama UML da classe `OranAdaptativeSqlite` implementada.

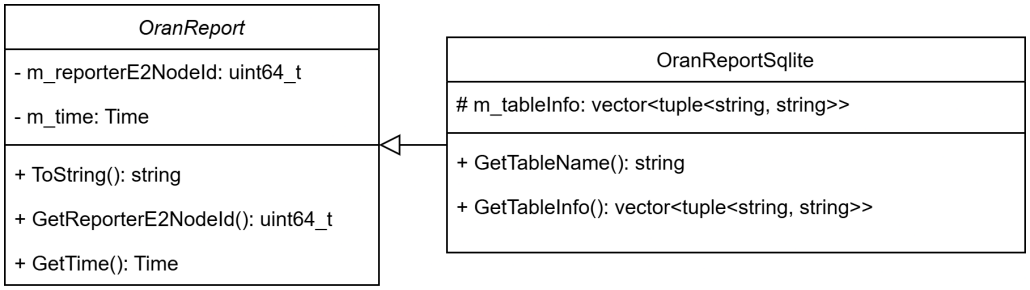

Figura 2. O diagrama UML da classe `OranReportSqlite` implementada.

na documentação fornecida com a ferramenta. A Figura 2 exibe o diagrama UML da classe `OranReportSqlite`.

A Figura 3 ilustra as diferenças entre o sistema implementado nesta versão e o original. No original, as tabelas padrão são criadas para cada relatório implementado, não sendo possível adicionar novas informações ou receber relatórios personalizados. A versão proposta permite a criação de relatórios personalizados, além da criação de tabelas novas à medida que relatórios de novos tipos são recebidos pelo Near-RT RIC. Na Figura 3, as tabelas criadas dinamicamente, bem como os relatórios personalizados, estão indicados em verde.

4. Casos de Uso

Para ilustrar a utilização das funcionalidades implementadas, foram criados dois casos de uso: o `simple-db-example` e o `rl-handover-example`. O `simple-db-example` é um caso de uso simples que ilustra a utilização do novo sistema de banco de dados e criação de relatórios personalizados. Esse caso de uso também demonstra o funcionamento de relatórios padrão juntamente com relatórios personalizados simultaneamente. O `rl-handover-example` é um caso de uso mais complexo que ilustra como a ferramenta pode ser utilizada para coletar informações para o treinamento de uma aplicação de aprendizado por reforço para o gerenciamento de *handover*.

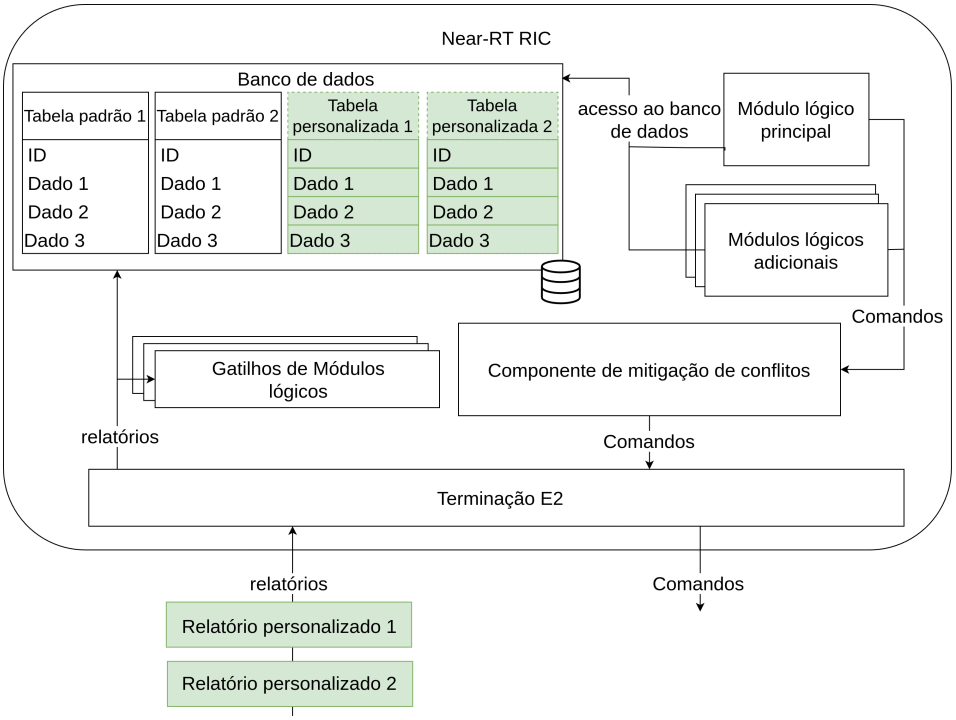

Figura 3. Near-RT RIC modificado. Em verde, os relatórios personalizados e as tabelas geradas dinamicamente.

4.1. Caso de Uso 1: Utilização de Relatórios Personalizados

Este caso de uso ilustra a criação de um relatório personalizado que reporta o endereço IPv4 de um UE periodicamente. O intuito é evidenciar a criação de relatórios e a implementação do novo sistema de banco de dados personalizado no *script* de simulação. O exemplo também mostra o funcionamento de relatórios personalizados juntamente com relatórios padrão do ns3-oran.

A Figura 4 ilustra o cenário implementado, no qual um equipamento de usuário conectado a um eNB envia periodicamente ao Near-RT RIC relatórios padrão, contendo sua localização, e personalizado, contendo seu endereço IPv4. No momento que o Near-

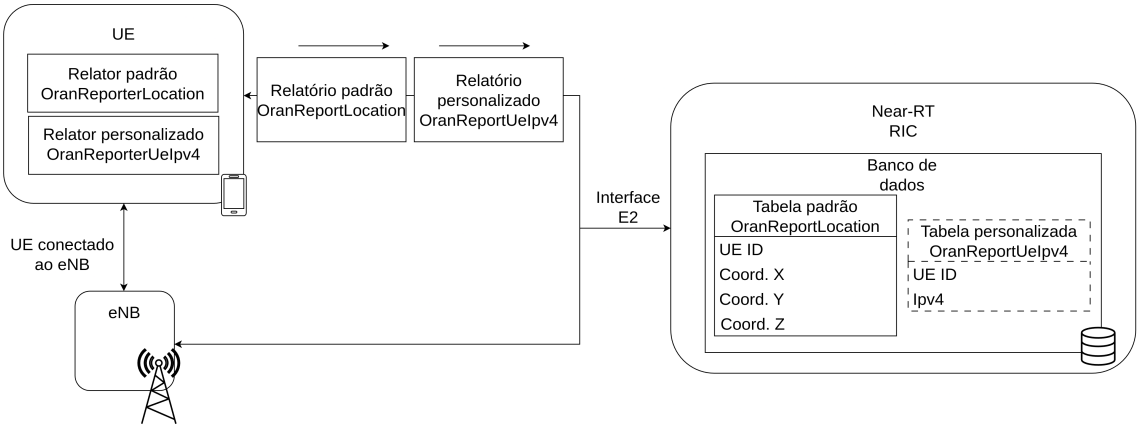

Figura 4. Cenário simulado em `simple-db-example`.

RT RIC recebe o relatório personalizado, uma nova tabela é gerada em seu banco de dados com base nos metadados contidos no relatório. A Lista 1 mostra a saída da execução do exemplo, imprimindo na tela as *queries* realizadas pelo Near-RT RIC ao banco de dados e o tempo de simulação. Na lista, é possível ver o momento em que a tabela do relatório personalizado é criada no bando de dados.

Lista 1. Exemplo de logs no banco de dados.

```
1 Query OK(101): "INSERT INTO node (nodetype) VALUES (?);" (1)
1 Query OK(101): "INSERT INTO noderegistration (nodeid, registered, simulationtime)
    VALUES (?, ?, ?);" (5, 1, 1000000000)
1 Query OK(101): "INSERT OR REPLACE INTO lteue (nodeid, imsi) VALUES (?, ?);" (0, 1)
2 Query OK(101): "SELECT registered FROM noderegistration WHERE nodeid = ? ORDER BY
    simulationtime DESC, entryid DESC LIMIT 1;" (5)
2 Query OK(101): "INSERT INTO nodelocation (nodeid, x, y, z, simulationtime) VALUES (?,
    ?, ?, ?, ?);" (5, 15, 0, 1.5, 2000000000)
2 Query OK(101): "CREATE TABLE IF NOT EXISTS UeIpv4 (entryid INTEGER PRIMARY KEY
    AUTOINCREMENT NOT NULL, nodeid INTEGER NOT NULL, time INTEGER NOT NULL, ipv4
    TEXT NOT NULL, FOREIGN KEY(nodeid) REFERENCES node(nodeid) );"
2 Query OK(101): "INSERT INTO UeIpv4 (nodeid, time, ipv4) VALUES (5, 2000000000,
    "7.0.0.2");"
3 Query OK(101): "SELECT registered FROM noderegistration WHERE nodeid = ? ORDER BY
    simulationtime DESC, entryid DESC LIMIT 1;" (5)
3 Query OK(101): "INSERT INTO nodelocation (nodeid, x, y, z, simulationtime) VALUES (?,
    ?, ?, ?, ?);" (5, 20, 0, 1.5, 3000000000)
3 Query OK(101): "INSERT INTO UeIpv4 (nodeid, time, ipv4) VALUES (5, 3000000000,
    "7.0.0.2");"
4 Query OK(101): "SELECT registered FROM noderegistration WHERE nodeid = ? ORDER BY
    simulationtime DESC, entryid DESC LIMIT 1;" (5)
4 Query OK(101): "INSERT INTO nodelocation (nodeid, x, y, z, simulationtime) VALUES (?,
    ?, ?, ?, ?);" (5, 25, 0, 1.5, 4000000000)
4 Query OK(101): "INSERT INTO UeIpv4 (nodeid, time, ipv4) VALUES (5, 4000000000,
    "7.0.0.2");"
```

4.2. Caso de Uso 2: *Handover* com Aprendizado por Reforço

Este caso de uso ilustra como a ferramenta pode ser utilizada no treinamento de um modelo para aprendizado por reforço. O exemplo simula um cenário simplificado, o cenário possui dois eNBs inicializados em posições fixas, nas extremidades do cenário, e um UE inicializado em uma posição aleatória em uma área entre os dois eNBs. Os eNBs e o UE são inicializados em posições colineares, o UE então se move em direção ao eNB mais distante. Durante a simulação o UE envia relatórios ao Near-RT RIC de taxa de perda de pacotes, localização atual e relação sinal-ruido através do eNB conectado. Sempre que o Near-RT RIC recebe um relatório de taxa de perda de pacotes que indique mais de 10% de perda, uma observação é enviada a um modelo sendo treinado com aprendizado por reforço, utilizando o algoritmo *proximal policy optimization – PPO*, em um programa em Python. O programa em Python retorna um comando de ação, indicando se um *handover* deve ser realizado e em qual eNB o UE deve se conectar. O cenário de simulação é repetido diversas vezes durante o treinamento do modelo, podendo ser interrompido pelo usuário. Quando o modelo chega a um desempenho satisfatório, é possível interromper o treinamento e utilizar o modelo apenas para inferência no gerenciamento de *handover*. Esse exemplo utiliza a extensão ns3-ai [Yin et al. 2020] para a comunicação entre o modelo sendo treinado em Python e o ns-3. A Figura 5 mostra a evolução do modelo com o tempo. Na figura, cada ponto corresponde à recompensa média recebida pelo modelo em um episódio no cenário de treino. A recompensa foi definida como $1 - p$, onde p é a taxa de perda de pacotes do UE simulado.

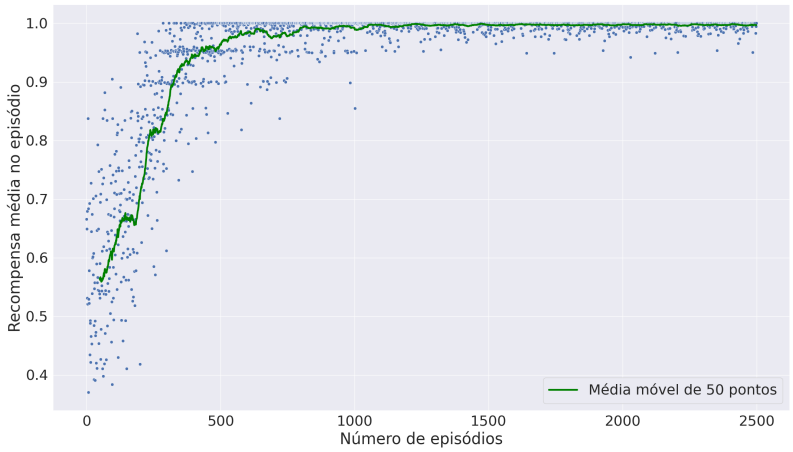

Figura 5. Evolução do treinamento do modelo de aprendizado por reforço para *handover* no cenário de exemplo.

Maiores informações sobre o exemplo, bem como uma ilustração do cenário de simulação podem ser encontradas na documentação disponibilizada com a ferramenta.

5. Conclusão e Trabalhos Futuros

A ferramenta desenvolvida impulsiona o desenvolvimento de aplicações baseadas em dados para o controle e diagnóstico da rede de acesso. Com as adições feitas, é possível utilizar o simulador como cenário para treinamento de aplicações de aprendizado de máquina e, em especial, para o treinamento de aplicações de aprendizado por reforço. Os componentes de banco de dados implementados dão liberdade ao usuário para coletar, armazenar e recuperar métricas arbitrárias. Isso permite a criação de cenários de simulação mais complexos e realistas, sendo útil em especial no treinamento de modelos a partir de cenários de simulação do ns-3. Uma versão desta ferramenta com suporte à integração com a extensão 5G-LENA [Patriciello et al. 2019] está sendo desenvolvida, e pode ser encontrada como um *branch* no GitHub da ferramenta.

Referências

ns-3 (2025). `https://www.nsnam.org/`. `https://www.nsnam.org/`. [Accessed 14-02-2025].

ns3-oran (2025). `https://github.com/usnistgov/ns3-oran`. [Accessed 14-02-2025].

O-RAN Alliance (2025). *`https://www.o-ran.org/`*. [Accessed 14-02-2025].

Patriciello, N., Lagen, S., Bojovic, B., and Giupponi, L. (2019). An e2e simulator for 5g nr networks. *Simulation Modelling Practice and Theory*, 96:101933.

Polese, M., Bonati, L., D'Oro, S., Basagni, S., and Melodia, T. (2023). Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials*.

Yin, H., Liu, P., Liu, K., Cao, L., Zhang, L., Gao, Y., and Hei, X. (2020). Ns3-ai: Fostering artificial intelligence algorithms for networking research. In *Proceedings of the 2020 Workshop on Ns-3*, WNS3 2020, page 57–64, New York, NY, USA. Association for Computing Machinery.