



Inteligência Artificial/Aprendizado de
Máquina

**Investigação de soluções
inteligentes para OpenRAN**

M3 - A3.2

Programa OpenRAN@Brasil - Fase 2

Sumário

Lista de ilustrações	3
Lista de tabelas	5
1 Introdução	6
2 Repositórios de dados para treinamento de AI/ML	7
2.1 Dados da rede Open RAN em construção na Fase 1	7
2.2 Simuladores para geração de dados para AI/ML em O-RAN	8
2.3 Trabalhos Prévios Usando ns-3 para Open RAN	11
2.4 Atuais Limitações da Solução Pautada no ns-3 com Módulo Open RAN	13
2.5 NORI: Nova Interface Open RAN para ns-3	15
2.6 Características dos Canais Sem fio do ns-3 de Importância ao Projeto	17
2.7 Formatos dos dados	22
3 Soluções baseadas em AI/ML	24
3.1 Organização	24
3.2 xApp e rApp Desenvolvidas	24
3.3 Gerenciamento de recursos de rádio (RRM) para fatiamento da rede	26
3.4 Aprendizado por Reforço Aplicado a Posicionamento de VNFs	30
4 Conclusão	33
5 Referências	34
6 Histórico de versões deste documento	36
7 Execução e aprovação	37

Lista de ilustrações

Figura 1 – Simuladores <i>application-specific</i> (<i>Asim</i>) e <i>generic</i> (<i>Gsim</i>) de RAN. Este projeto desenvolve um <i>Gsim</i> com base no ns-3.	8
Figura 2 – SD-RAN RANSIM (RAN <i>simulator</i>) arquitetura da < https://docs.sdt-ran.org/master/ran-simulator/README.html >.	10
Figura 3 – Arquitetura do ns-O-RAN descrito em Lacava et al. [1].	11
Figura 4 – Arquitetura do software desenvolvido na UnB e descrito em Ferreira et al. [2].	12
Figura 5 – Descrição do cenário simulado em Lacava et al. [1], com versão do ns-3 que não dá suporte à interferência <i>intercell</i>	15
Figura 6 – Diagrama simplificada da implementação realizada.	20
Figura 7 – Ambiente no Sionna gerado via <i>OpenStreetMap</i>	21
Figura 8 – SINR calculada ao longo do tempo e da frequência de transmissão da portadora.	21
Figura 9 – Métricas de desempenho para cada frequência de transmissão da portadora.	22
Figura 10 – Versão resumida da arquitetura de componentes do <i>Near-RT RIC</i>	26
Figura 11 – Taxa de transferência obtida por cada fatia de rede pelo método proposto e o <i>baseline</i> SSR. A fatia de URLLC possui requisito de 5 Mbps e a fatia de eMBB 20 Mbps. A fatia de mMTC não possui requisito definido para taxa de transferência.	28
Figura 12 – Latência no buffer média para cada fatia de rede. A fatia de URLLC, eMBB e mMTC possuem requisitos de 1 ms, 30 ms e 50 ms.	29

Figura 13 – Número de violações totais e nas fatias de URLLC ao utilizar os métodos de SSR e o método proposto.	29
Figura 14 – Exemplo de uma topologia vRAN com <i>clusters</i> que utilizam a técnica de CoMP para mitigação de interferência.	31
Figura 15 – Como a solução proposta se encaixa na arquitetura O-RAN.	31
Figura 16 – Eficiência espectral e demanda por recursos computacionais sobre diferentes condições de carga na rede.	32

Lista de tabelas

Tabela 1 – Requisitos para o simulador genérico NORI em desenvolvimento, onde Prior. indica prioridade dada pela equipe do projeto, com “-” indicando não fazer parte do escopo do projeto.	15
---	----

1 Introdução

Este relatório corresponde a um dos entregáveis do Projeto Open-RAN@Brasil desenvolvido em parceria entre a Rede Nacional de Ensino e Pesquisa (RNP), Universidade Federal de Goiás (UFG), Universidade Federal do Pará (UFPA) e Universidade do Vale do Rio dos Sinos (UNISINOS).

Objetivos do Relatório

O objetivo deste documento é descrever:

- as atividades realizadas no âmbito da Atividade 3.2;
- as decisões tomadas e que alicerçam essas atividades;
- os resultados preliminares alcançados;
- recursos já disponibilizados à comunidade Open RAN brasileira.

2 Repositórios de dados para treinamento de AI/ML

O repositório ainda não se encontra construído. A falta de disponibilidade da plataforma Open RAN em desenvolvimento na Fase 1 do projeto, não permite ainda a obtenção de dados e medidas oriundos desta plataforma. Por outro lado, as atividades pautadas em simuladores de redes de acesso por rádio (RAN) encontram-se avançadas, e são a tônica deste relatório.

Registra-se que apesar de ainda não concluído, alguns aspectos do repositório já estão definidos pela equipe, com o mesmo sendo baseado na experiência do grupo na construção e disponibilização de recursos para AI/ML.

Todos os dados e componentes de software estão sendo construídos em repositório temporário no LABORA, em <<https://github.com/LABORA-INF-UFG>>.

2.1 Dados da rede Open RAN em construção na Fase 1

Os dados da rede de comunicações Open RAN em construção na Fase 1 do projeto ainda não estão disponíveis, pois houve atraso na execução do cronograma. Enquanto isso, a equipe do projeto vem aperfeiçoando simuladores, os quais estão sendo usados e customizados para geração de dados para uso em AI/ML.

2.2 Simuladores para geração de dados para AI/ML em O-RAN

Pode-se organizar as ferramentas que dão suporte à simulação da RAN no contexto Open RAN em duas categorias: as *application-specific* (*Asim*) e as *generic* (*Gsim*). Os simuladores *Asim* são concebidos para aplicações e simulações específicas de um dado caso de uso, enquanto os *Gsim* são genéricos, dando suporte a uma ampla gama de simulações. A Figura 1 ilustra esses dois tipos de simuladores de RAN e ambos são discutidos neste relatório. Como será detalhado adiante, a equipe do projeto vem desenvolvendo um *Gsim*, a partir da adaptação do simulador de redes ns-3.

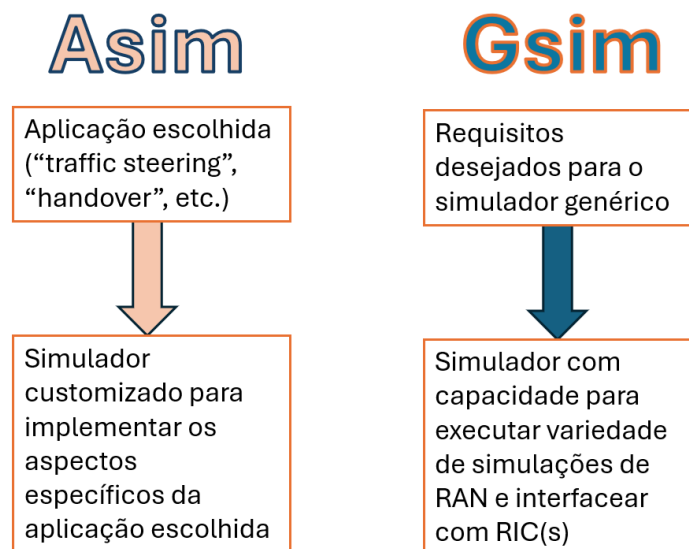


Figura 1 – Simuladores *application-specific* (*Asim*) e *generic* (*Gsim*) de RAN. Este projeto desenvolve um *Gsim* com base no ns-3.

O termo “simulador” é usado em distintos contextos, e mesmo se a atenção estiver restrita a simuladores de redes de comunicações, há distintas categorias de simuladores. No contexto de Open RAN, há diversos simuladores, mas é importante que se faça distinção de seus objetivos. Uma rede de comunicações móveis que siga um padrão do grupo 3GPP (4G, 5G, etc.) é composta pelo núcleo, rede de transporte e RAN. Neste contexto, Open RAN dedica especial atenção à RAN. Dessa forma, os simuladores na categoria *Gsim* de redes de comunicações adequados para investigações de AI/ML no Open RAN são exatamente os que propiciam suporte amplo a simulações da RAN.

Dentre os simuladores específicos da comunidade Open RAN que foram avaliados pela equipe do projeto ao longo da Atividade 3.2, encontram-se o RAN Simulator (ou RANSIM, descrito em <https://docs.sd-ran.org/master/ran-simulator/README.html>), que é parte da plataforma SD-RAN (*Software-Defined RAN*) desenvolvida pela *Open Networking Foundation* (ONF), e o E2 Simulator (<https://wiki.o-ran-sc.org/display/SIM/E2+Simulator>) da O-RAN *Software Community* (SC), que é uma colaboração entre a O-RAN ALLIANCE e a Linux Foundation.

O E2 Simulator foi concebido para testes dos *protocolos e mensagens* relacionados à interface E2 da especificação O-RAN. A atual versão do mesmo não se propõe a fazer simulação dos aspectos de transmissão da informação via rádio na RAN em si, ou seja, não incorpora modelos de canais de propagação de rádio, etc. O RAN Simulator da SD-RAN pode ser categorizado como um *Gsim*, e tem a arquitetura descrita na Figura 2. Este software foca na simulação de nós CU/DU e células definidas por RUs, através do padrão O-RAN E2AP. Como indicado na Figura 2, o RANSIM define internamente os códigos que representam UEs e gNBs. Com isso, a interface do RANSIM com o mundo externo não deve reutilizar implementações de tais nós da rede. Por exemplo, a arquitetura do RANSIM não foi concebida para que o mesmo tenha interface com um simulador como o ns-3, que também esteja provendo código para UEs e gNBs se comunicarem.

O bloco *Radio Emulation* na Figura 2 (em cor laranja) auxilia a simulação da RAN, a qual é complementada pela simulação da mobilidade dos UEs, etc. Por exemplo, o código em <https://github.com/onosproject/ran-simulator/blob/9db3dd396bef9b0eaaadc325b0de2b616dca27b4/pkg/mobility/signal.go#L56> possui funções para cálculo da distância Eucliana entre transmissores e receptores, assim como cálculos simples da perda de percurso (*path loss*). A partir de tais cálculos, o RANSIM pode implementar algoritmos para *handover*, com métodos como implementado em <https://github.com/onosproject/ran-simulator/blob/9db3dd396bef9b0eaaadc325b0de2b616dca27b4/pkg/mobility/driver.go#L374>.

Como desenvolvido *from scratch* para Open RAN e integrado aos demais com-

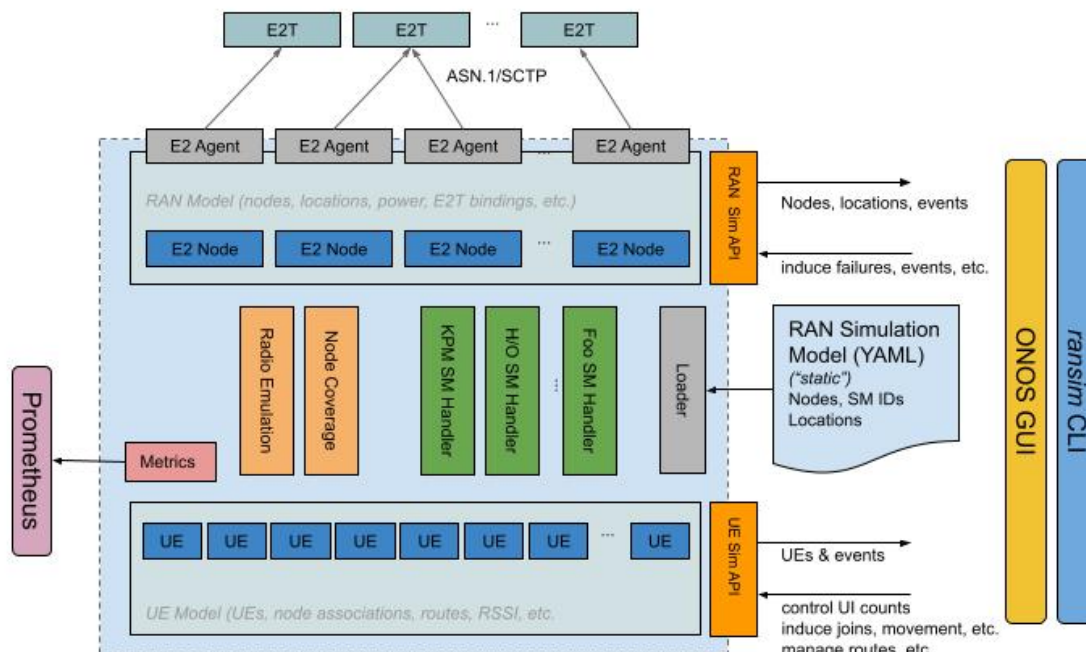


Figura 2 – SD-RAN RANSIM (RAN *simulator*) arquitetura da <https://docs.sd-ran.org/master/ran-simulator/README.html>.

ponentes de software da SD-RAN, o RANSIM agrega bastante código que implementa especificações Open RAN. Por exemplo, o RANSIM fornece suporte para ASN.1 e também *service models* (SM, vide <https://docs.sd-ran.org/master/onos-e2-sm/README.html>) como *Key Performance Metrics* (E2SM_KPM), *RAN Control* (E2SM_RC_PRE) e *RAN Slicing* (E2SM_RSM). Entretanto, nem todos SMs estão implementados completamente e de acordo com a última especificação, o que é típico de uma tecnologia, como Open RAN, quando está sendo definida e em franca expansão.

Apesar de ter amplo código para Open RAN, o RANSIM não tem código para simular canais de comunicação sem fio e algoritmos relacionados a telecomunicações. Por exemplo, as métricas que o RANSIM oferece via E2 são de “alto nível”, tal como a contagem de UEs conectados (vide <https://github.com/onosproject/ran-simulator/blob/9db3dd396bef9b0eaaadc325b0de2b616dca27b4/pkg/servicemodel/kpm2/defs.go>). De forma oposta, consagrados simuladores de eventos discretos aplicados a redes de

comunicações como ns-3 [3] e Omnet++ não incorporam suporte a Open RAN em seus *codebases*. De forma a diminuir esse *gap*, os autores de Lacava et al. [1] estenderam o ns-3 para que dê algum suporte a Open RAN. Tal estratégia é o assunto das próximas seções.

2.3 Trabalhos Prévios Usando ns-3 para Open RAN

A arquitetura adotada em Lacava et al. [1] está descrita na Figura 3 e o código foi incorporado pela SC, que o disponibiliza em <https://github.com/o-ran-sc/sim-ns3-o-ran-e2>. Uma iniciativa semelhante foi desenvolvida na UnB [2], com código disponibilizado em https://gabrielcarvfer.github.io/NS3/ns3_ORAN/. Ambas concernem o suporte a xApps executados pelo near-real-time RIC. Não há resultados oriundos de rApps, mas o mesmo é citado como um possível trabalho futuro em Ferreira et al. [2].

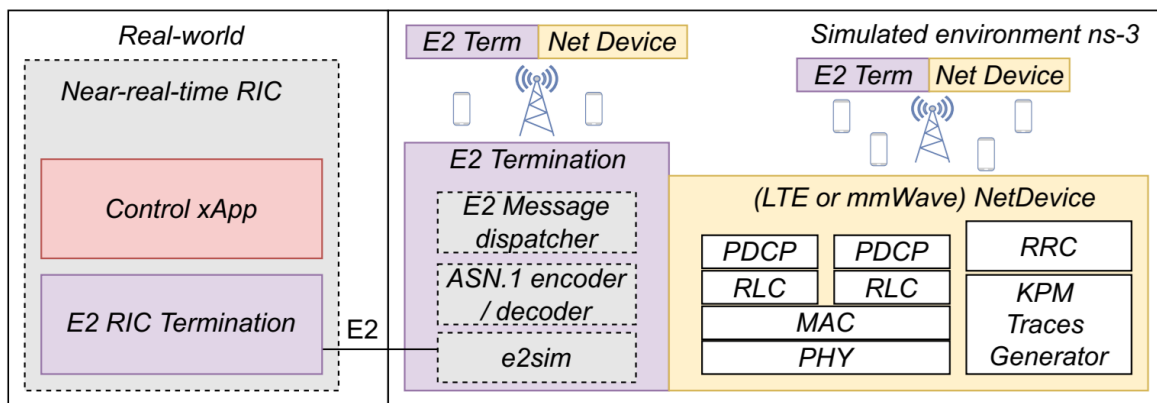


Figura 3 – Arquitetura do ns-O-RAN descrito em Lacava et al. [1].

O código ns-O-RAN disponibilizado em Lacava et al. [1] basicamente suporta a interface E2 e implementa dois E2SMs diferentes: E2SM KPM para relatórios e o E2SM RAN *Control* (RC), o qual habilita o RIC a impor ações de controle à RAN (por exemplo, *traffic steering* e *mobility management*). O ns-O-RAN é estruturado como um módulo externo do ns-3 e usa SCTP para conectar o simulador e o near-RT RIC. Contudo, mesmo que estruturado como um módulo externo do ns-3, os autores se permitiram modificar

código do próprio ns-3, inviabilizando a compatibilidade do módulo desenvolvido com novas versões do ns-3.

Diferentemente de Lacava et al. [1], a ferramenta proposta em Ferreira et al. [2] implementa o near-RT RIC e os xApps a partir de um mesmo *codebase*, usando a mesma linguagem de programação, e visando execução em uma só máquina. De acordo com os autores de Ferreira et al. [2], foram implementados parcialmente 5 dos 12 procedimentos elementares do E2AP e 11 dos 27 tipos de mensagens previstos para os procedimentos implementados. Estes são suficientes para a implementação de *loops* de controle com um único near-RT RIC e topologia estática da RAN.

Em Ferreira et al. [2], os autores optaram por implementar o near-RT RIC na classe do ns-3 responsável pelo *gateway* SGW, conforme ilustra a Figura 4. Para demonstrar a funcionalidade da ferramenta e verificar seu funcionamento, foi implementado o serviço de *handover* incondicional, em conjunto com as métricas requeridas para a tomada de decisões, como parte do *Service Model* E2 de Controle da RAN (E2SM-RC).

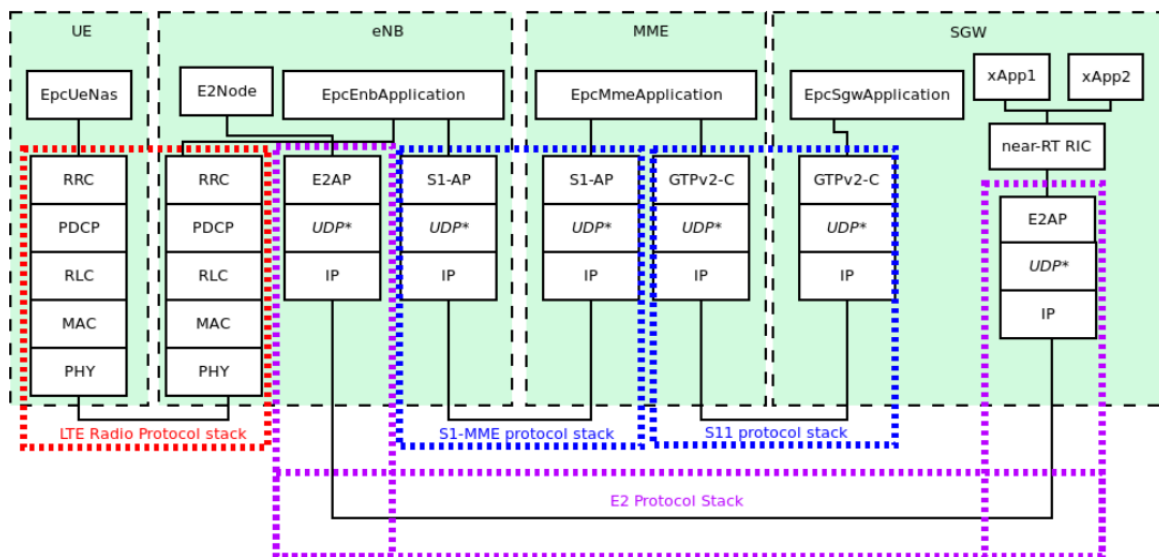


Figura 4 – Arquitetura do software desenvolvido na UnB e descrito em Ferreira et al. [2].

Contrastando-se as Figuras. 3 e 4 com a Figura 2, pode-se observar as distintas demandas de aperfeiçoamentos desses três simuladores em prol de seu uso para

simular Open RAN. Enquanto que UEs e gNBs já possuem suporte intrínscico à Open RAN na implementação associada à Figura 2, as Figuras. 3 e 4 revelam que cada nó da simulação ns-3 que precise se comunicar via E2, requer mudança de seu código ns-3 para que o nó se comporte como um E2 *Terminator*. Contudo, assume-se que não há muitos elementos da RAN do ns-3 que exijam tal atualização do código. Assim, após revisão do estado-da-arte, e dentre as opções de simuladores *Gsim* que estão sendo desenvolvidas, a equipe do projeto entendeu que a estratégia mais promissora é estender o ns-3, na linha defendida por Lacava et al. e Ferreira et al. [1, 2].

2.4 Atuais Limitações da Solução Pautada no ns-3 com Módulo Open RAN

Dada a decisão de projeto de expandir o ns-3 para dar suporte ao Open RAN, é adequado avaliar quais as limitações dos trabalhos descritos em Lacava et al. e Ferreira et al. [1, 2] e quais ações devem ser tomadas pela equipe do projeto para complementá-los de maneira a atingir os objetivos do projeto Open-RAN@Brasil.

A primeira questão crítica associada a ambos trabalhos [1, 2] diz respeito à modularidade. Ambos fazem modificações no código do ns-3 e disponibilizam componentes de software que são potencialmente incompatíveis com novas versões do ns-3. Similar a este aspecto, em Lacava et al. [1], os autores usam versão modificada do FlexRIC e não se alinham com a constante evolução dos RICs da SC, por exemplo.

No importante aspecto da modelagem do canal sem fio, o software descrito em Lacava et al. [1] não busca se manter alinhado com as novas versões do ns-3, pois utiliza um módulo para ondas milimétricas que é customizado pelos autores de Lacava et al. [1] e não integrado ao *main branch* do ns-3.

Outro aspecto a ser considerado é que o ns-3, independente do módulo Open RAN, dá limitado suporte à desagregação dos elementos da RAN. Por exemplo, não há no ns-3, nós CU, DU e RU. Este problema específico da desagregação foge do escopo da

atuação da equipe, mas é de relevância ao projeto.

Outra restrição do próprio simulador ns-3 refere-se ao suporte a técnicas com múltiplas antenas. Para melhor entender, deve-se considerar que atualmente a maior parte das simulações de 5G com ns-3 dependem não apenas do código do ns-3, mas do módulo 5G-Lena. Especificamente, até a versão 40 do ns-3, as técnicas para o processamento de sinais trafegando em canais sem fio não usavam algoritmos avançados de MIMO, tal como multiplexação espacial. Nessas versões mais antigas, o uso de algoritmos para múltiplas antenas estava restrito pelo fato do ns-3 dar suporte a apenas uma porta no UE e na gNB (com suporte a apenas uma camada). A versão 40 iniciou suporte a processamento MIMO com transmissores e receptores dotados de múltiplas portas, o que é essencial para 6G e importante para alguns casos de uso do Open RAN.

Outra evolução pela qual o ns-3 e o módulo 5G-Lena (NR) estão atualmente passando, refletida nas versões 42 (de maio de 2024) do ns-3 e 3.0 do 5G-Lena, é a modelagem da interferência *intercell* (ou *out-of-cell*), levando em consideração características fundamentais de uma transmissão MIMO com múltiplas portas. Por exemplo, na versão adotada em Lacava et al. [1], a simulação descrita pela Figura 5 não modelava interferência *intercell* com essas características. Dessa forma, apesar da Figura 5 indicar existência de quatro estações rádio-base, não há qualquer interação dos sinais interferentes das mesmas. Elas existiram na simulação em Lacava et al. [1] apenas para indicar qual era o sinal mais forte chegando ao UE dentre os gerados pelas quatro estações rádio-base.

Essa restrição não permite, por exemplo, a avaliação usando ns-3 + O-RAN de algoritmos de IA/ML para aumento da capacidade do 6G através da minimização da interferência entre as células. Este é um exemplo da eventual limitação na simulação de xApps e rApps usando a ferramenta no seu atual estágio. Isso motivou o esforço em prol de melhorias na adoção do ns-3 para Open RAN, como descrito a seguir.

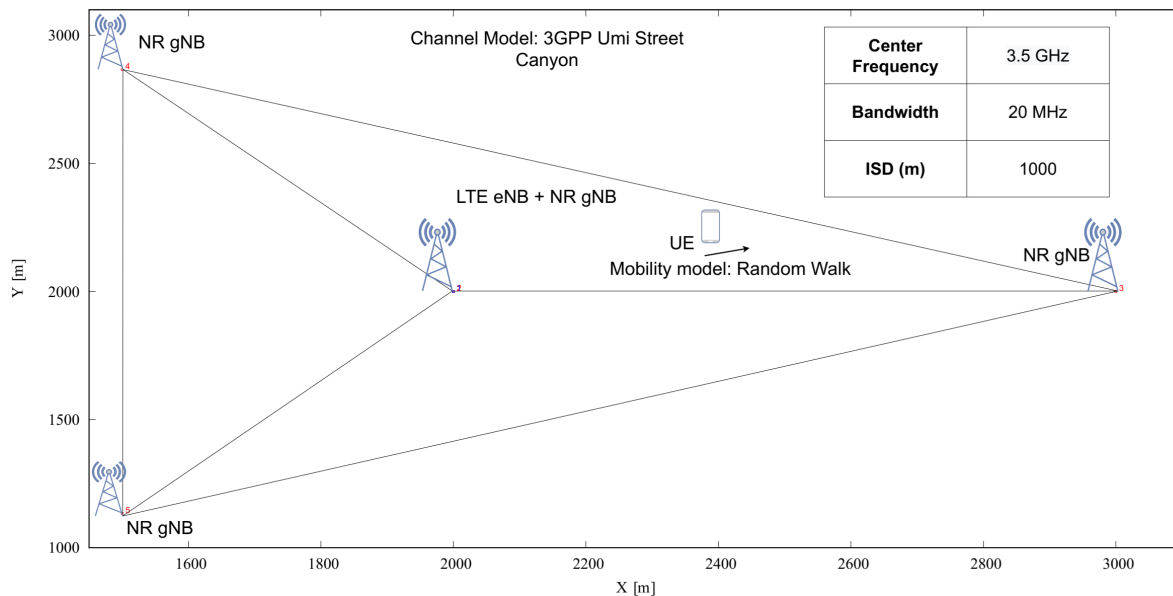


Figura 5 – Descrição do cenário simulado em Lacava et al. [1], com versão do ns-3 que não dá suporte à interferência *intercell*.

2.5 NORI: Nova Interface Open RAN para ns-3

Com base nos objetivos almejados e limitações citadas, a Tabela 1 apresenta os principais requisitos para o simulador da categoria *Gsim* em desenvolvimento neste projeto, chamado de NORI.

Tabela 1 – Requisitos para o simulador genérico NORI em desenvolvimento, onde Prior. indica prioridade dada pela equipe do projeto, com “-” indicando não fazer parte do escopo do projeto.

#	Descrição do suporte desejado	Prior.
1	Compatibilidade contínua com o RIC da SC	2
2	Compatibilidade contínua com o ns-3 e módulo 5G-Lena	1
3	Inclusão e suporte eficaz a novos <i>service models</i>	5
4	Desagregação de elementos (<i>functional splits</i>) do ns-3	-
5	Canais sem fio com consistência no tempo, frequência e espaço	3
6	<i>Plug-and-play</i> de xApps NORI em redes O-RAN com RICs SC	4
7	<i>Plug-and-play</i> de xApps NORI em redes O-RAN com RICs comerciais	-
8	Algoritmos MIMO envolvendo múltiplas camadas	-
9	<i>Plug-and-play</i> de rApps NORI em redes O-RAN com RICs SC	-

O NORI (*New Open RAN Interface*) é um módulo externo desenvolvido para

o simulador de redes ns-3, com o objetivo de conectar cenários do ns-3 à arquitetura Open RAN. Posicionado como uma extensão do trabalho de Lacava et al. [1], o NORI foi projetado para funcionar em conjunto com a interface *oran-interface* (apresentada em Lacava et al. [1]) e com o módulo ns-3 para *New Radio* (NR) chamado 5G-Lena (<<https://5g-lena.cttc.es/>>). Isso permite que NORI seja mais facilmente compatível com novas versões do ns-3 e do 5G-Lena do que seus precursores.

A tarefa de portar código de Lacava et al. [1] para o NORI consiste principalmente em entender e desfazer as modificações nas classes do ns-3 e mover as respectivas funcionalidades para um módulo externo que siga as regras de desenvolvimento do ns-3. Nessa ação, o projeto herda de Lacava et al. [1], por exemplo, os *Service Models* KPM e RC. Entretanto, para SMS ainda não desenvolvidos, será usada metodologia distinta, ajustada às novidades que estão sendo incorporadas aos ecossistemas ns-3 e Open RAN. Para que o NORI funcione corretamente com os módulos NR e *oran-interface*, foram criadas classes customizadas dentro do módulo NORI. Essas classes são responsáveis pelo gerenciamento das gNBs e UEs. A arquitetura NORI segue a descrita em Lacava et al. [1], e inclui elementos que permitem a comunicação e controle entre os componentes de rede simulados e as interfaces definidas pela O-RAN.

Como indica o requisito de índice 2 da Tabela 1, o NORI está sendo desenvolvido para garantir compatibilidade com as novas versões do ns-3 e do módulo NR, assegurando que esteja sempre atualizado com as últimas inovações e melhorias nos dois *codebases*. Isso inclui suporte a novos recursos, atualizações de protocolo e melhorias de desempenho, que são regularmente introduzidas no ecossistema ns-3.

Alguns dos requisitos listados na Tabela 1 são pautados em uma engenharia de software em prol de modularidade, compatibilidade e eficácia. Em contraste, o requisito de número 5 concerne a simulação de canais de rádio frequência e o 8 se refere ao suporte a algoritmos para MIMO. Ambos dependem primordialmente de processamento de sinais aplicado a telecomunicações. Nestes dois tópicos, os esforços da equipe do projeto estão alinhados com o desenvolvimento do suporte a 5G NR no ns-3 (vide <<https://>

[//gitlab.com/cttc-lena/nr/-/blob/5g-lena-v3.0.y/doc/source/nr-module.rst](https://gitlab.com/cttc-lena/nr/-/blob/5g-lena-v3.0.y/doc/source/nr-module.rst)>).

2.6 Características dos Canais Sem fio do ns-3 de Importância ao Projeto

Esta seção descreve alguns aspectos relacionados à simulação dos canais sem fio que são relevantes ao projeto, iniciando pela categorização de técnicas de simulação.

Dependendo do caso de uso, há dois tipos de simulações de interesse para avaliação de AI/ML em Open RAN: simulações em nível de sistema (*system level*) versus simulações em nível de enlace (*link level*). Os simuladores de rede em nível de sistema permitem uma avaliação fim-a-fim (*end-to-end*) de algoritmos e protocolos, e são fundamentais quando a aplicação de AI/ML no Open RAN requer tal avaliação fim-a-fim do sistema.

A adequação dos simuladores para essas tarefas depende em grande parte da precisão do modelo de canal e da escalabilidade do mesmo para implementações de escala realista (por exemplo, no número de terminais móveis). Os simuladores em nível de sistema geralmente abstraem a simulação da transmissão efetiva de formas de onda ou de bits em nível de enlace, e apenas adotam um modelo de erro, que mapeia a razão sinal-interferência mais ruído (SINR) do enlace para uma probabilidade de erro de pacote. Contudo, a adoção de tais modelos de erro às vezes entram em conflito com a necessidade de simulação acurada dos canais sem fio. As tendências recentes das redes móveis atuais e futuras, que incluem grandes sistemas de antenas, implantações massivas e bandas de alta frequência, exigem modelos de canais complexos para a simulação precisa de MIMO massivo (m-MIMO) em ondas milimétricas (mmWave) e Terahertz (bandas THz). Ou seja, encontra-se alinhada com a evolução de Open RAN, o uso de frequências mais altas. Neste caso, as piores condições de propagação precisam ser compensadas pelo uso de enlaces direcionais (com técnicas de *beamforming*, etc.) assim como a densificação da rede.

Essa tendência tecnológica exige que o simulador adote modelos de canal que deem suporte às altas frequências, de forma que os valores de SINR sejam calculados adequadamente. Em particular, uma caracterização precisa do padrão de radiação da antena e do efeito da presença de múltiplos elementos radiantes torna-se extremamente importante no estudo das frequências mmWave e THz. Como resposta a essa problemática, o grupo 3GPP definiu o modelo de canal TR 38.901. O 3GPP TR 38.901 pode ser considerado um modelo híbrido. O mesmo é estocástico, mas também fornece consistência espacial, com a intenção de dar suporte adequado a simulações com múltiplas antenas (*beamforming*, multiplexação espacial em MIMO, etc.). O modelo 3GPP TR 38.901 foi implementado no simulador ns-3 por Tommaso Zugno [4] no Google *Summer of Code* de 2019, e está sendo continuamente melhorado pela comunidade do ns-3. A implementação original exigia alto custo computacional, como discutido a seguir.

Atualmente o ns-3 dá suporte a dois modelos de canal: 3GPP TR 38.901 e *two ray*. Em Pagin et al. [5], os autores descrevem uma melhoria na eficiência da modelagem de canal para simulações em nível de sistema MIMO em larga escala que tornou o ns-3 cinco vezes mais rápido na execução do modelo 3GPP TR 38.901, e o modelo *two ray* usa apenas 6% do tempo do 38.901. As extensões foram desenvolvidas em duas direções. Primeiro, os autores de Pagin et al. [5] melhoraram a eficiência do antigo código de implementação 3GPP TR 38.901 no ns-3. Em segundo lugar, propuseram um novo modelo de canal MIMO orientado para desempenho para complexidade reduzida, como um modelo alternativo adequado para bandas mmWave/THz, e o calibraram para gerar resultados semelhantes ao modelo 3GPP TR 38.901.

Uma alternativa ao modelo 3GPP TR 38.901 é o uso de traçado de raios para gerar canais sem fio. A equipe do projeto desenvolveu suporte no ns-3 ao uso dos resultados do software de traçado de raios Sionna da empresa NVIDIA. Este relatório apresenta resultados integrando o uso de ns-3 e de traçado de raio modelando um cenário urbano da cidade de Belém, Pará. O software Sionna é usado para calcular os canais de comunicação e, posteriormente, são conduzidas simulações fim-a-fim via ns-3 para

avaliar as métricas de desempenho, considerando a variação da frequência da portadora usada para transmissão.

Houve estudos com o ns-3 e traçado de raio, principalmente com a realização *quasi deterministic* [6]. No entanto, esses estudos não utilizaram os recursos mais recentes do ns-3, nem métodos de traçado de raio mais modernos, como o software de traçado de raio Nvidia Sionna. A equipe do projeto reimplementou o uso de traçado de raio no ns-3, desenvolvendo um novo módulo compatível com a versão mais atual do simulador e com interface via módulo NR. Esta reimplantação é demonstrada neste relatório com canais calculados pelo software Sionna em uma simulação 5G fim-a-fim para coletar métricas de desempenho e dados da camada física.

O cenário simulado é descrito a seguir. Na comunicação sem fio 5G *Non-Standalone* (NSA), a *Next Generation Node B* (gNB) conecta o *User Equipment* (UE) ao nó final que provê serviços de aplicação. A gNB se conecta ao nó final via provedor de serviço de Internet, utilizando uma conexão ponto-a-ponto. Em contraste, a comunicação entre a antena 5G e o UE é sem fio. Para replicar uma topologia fim-a-fim real, incluímos no cenário do ns-3 um UE, uma gNB conectada a um núcleo 4G (*Evolved Packet Core* - EPC) e um provedor de aplicação. Com o cenário fim-a-fim implementado no ns-3, realizamos a dinâmica da simulação conforme ilustrado na Figura 6. A implementação inicia com a simulação de traçado de raio, obtendo uma nova posição no cenário. Definimos um tempo de passo de 50 milissegundos (ms) para calcular e salvar o canal. Após cada computação, verificamos se o número de episódios atingiu o valor máximo, definido arbitrariamente como 1000, para decidir se a simulação continua ou é finalizada. Após calcular e salvar os canais, simulamos no ns-3 uma comunicação fim-a-fim, utilizando o canal calculado via Sionna. Com esse canal, computamos o sinal recebido e verificamos se o tempo de simulação foi excedido, o qual foi definido com base no tempo de passo, para finalizar ou não a simulação.

Para geração dos canais utilizados no simulador de redes, utilizamos um cenário tridimensional bem definido. O cenário escolhido foi urbano da cidade de Belém,

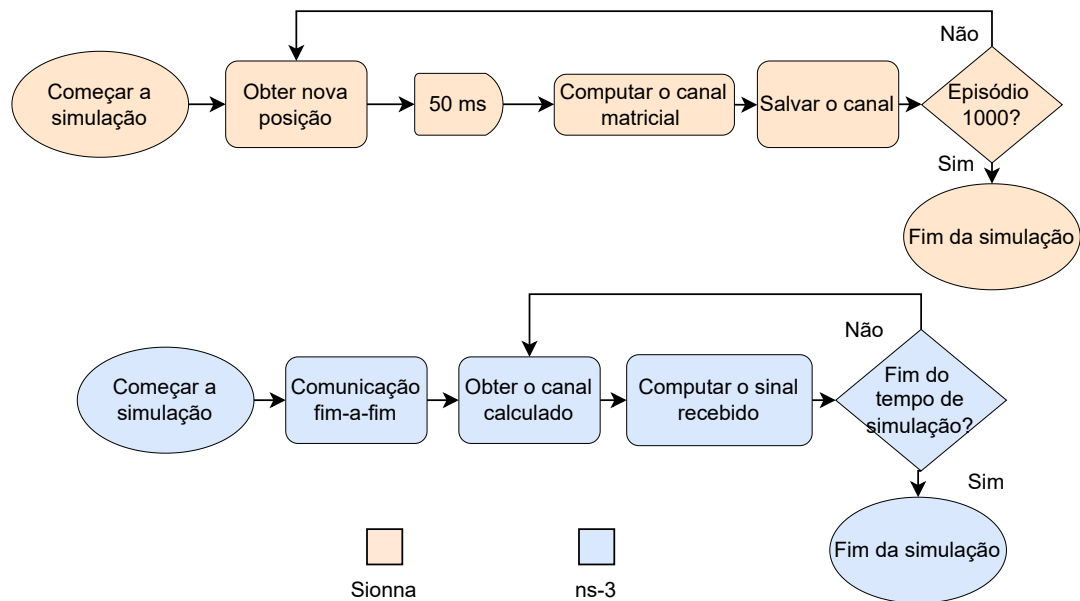
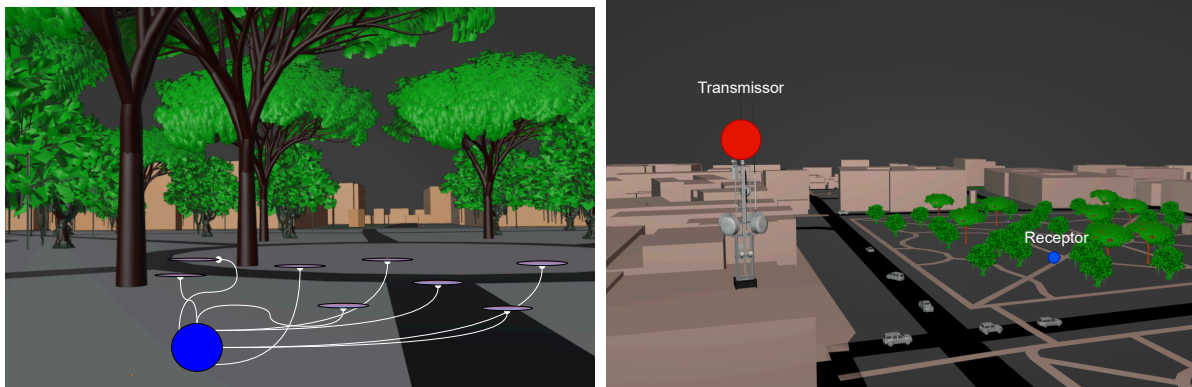


Figura 6 – Diagrama simplificado da implementação realizada.

mostrado na Figura 7, gerado com a ferramenta *OpenStreetMap*,¹ a qual permite criar modelos 3D para simulação. Com o cenário tridimensional criado, posicionamos os nós transmissor e receptor no cenário para replicar um ambiente real e fixamos a posição dos objetos presentes na cena (como carros, prédios, árvores, etc.), como visto na Figura 7b. O caminho percorrido aleatoriamente pelo nó receptor dentro da praça, mostrado na Figura 7a, permite a geração de novos canais para uso no simulador de redes, a cada nova posição. Além disso, realizamos 11 simulações, utilizando uma frequência da portadora variando de 20 GHz até 30 GHz.

Com os novos canais gerados pela simulação do Sionna, é possível realizar a simulação no ns-3, que obedecerá à mesma taxa de atualização de canais, definida pelo tempo de passo previamente configurado. Os resultados obtidos a partir da simulação no ns-3 foram extraídos utilizando os métodos `FlowMonitor()` e `EnableTraces()` do simulador, que permitem a obtenção de dados a nível de aplicação e camada física, respectivamente. As simulações tiveram duração de 50 segundos, com pacotes de 1024 bytes transmitidos em um intervalo de 1 ms.

¹ <https://www.openstreetmap.org>



(a) Caminho percorrido.

(b) Cenário 3D de Belém.

Figura 7 – Ambiente no Sionna gerado via *OpenStreetMap*.

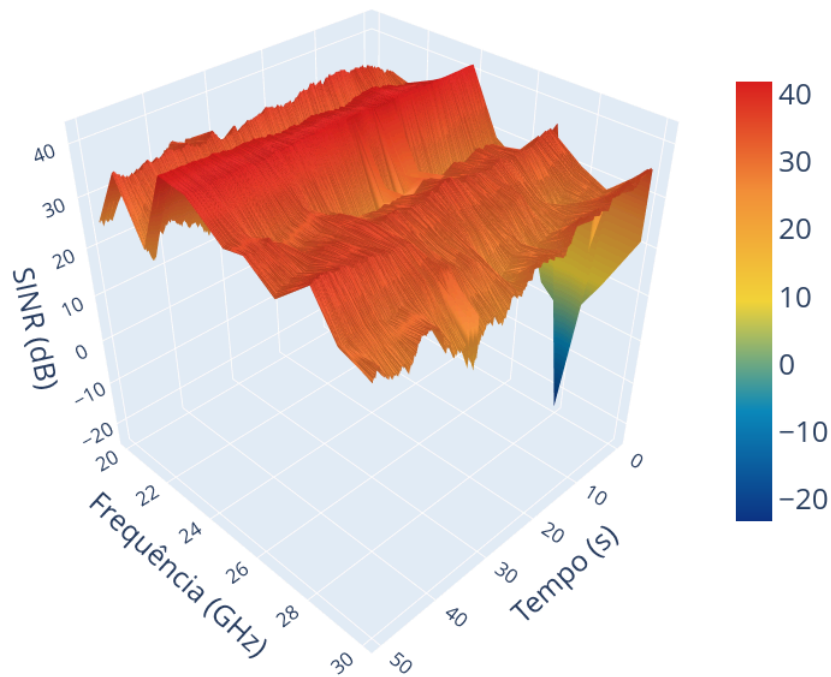


Figura 8 – SINR calculada ao longo do tempo e da frequência de transmissão da portadora.

Para este experimento, priorizamos a avaliação da taxa de SINR em dB na camada física para cada uma das 11 simulações realizadas, conforme ilustrado na Figura 8. Os resultados demonstram uma maior instabilidade em frequências mais altas, que sofrem maior atenuação durante a propagação e interação com o meio. Já para as frequências mais baixas, observa-se um maior grau de estabilidade, embora com algumas

flutuações intrínsecas à sua propagação e interação.

Apesar das flutuações no SINR, os resultados a nível de aplicação demonstram que o impacto nas métricas de desempenho se manteve estável, mesmo para frequências mais altas. Conforme a Figura 9, a média do *delay* não apresentou variações significativas em todas as simulações, mantendo-se entre 10 e 10,5 ms. De forma similar, a taxa de transferência (*throughput*) e o *delay jitter* se mantiveram estáveis dentro de uma faixa considerável, com aproximadamente 9,5 Mbps para a taxa de transferência e 1,7 ms para o *jitter*.

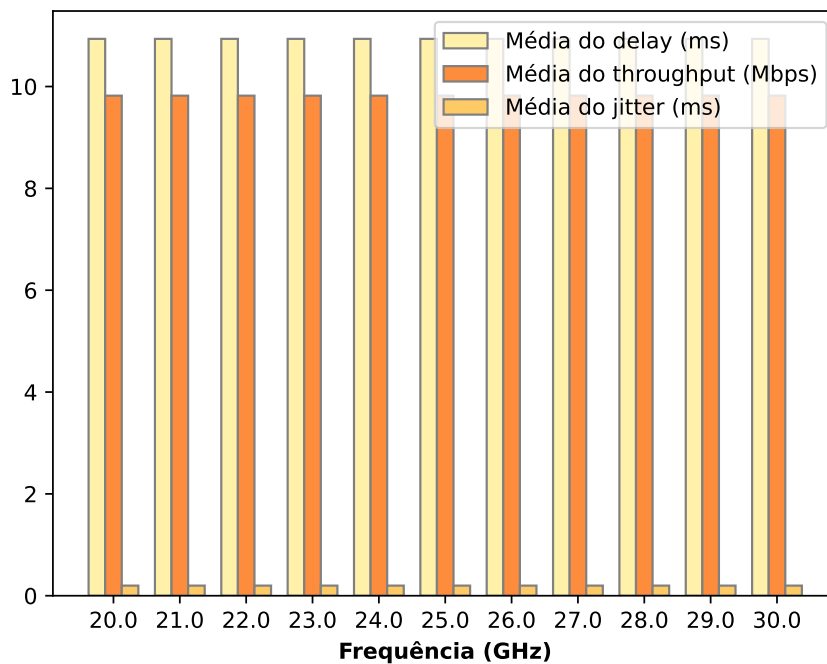


Figura 9 – Métricas de desempenho para cada frequência de transmissão da portadora.

2.7 Formatos dos dados

Esta seção sumariza algumas das discussões acerca da preparação do repositório, em especial a questão dos formatos dos dados.

Atualmente, já existem bases de dados de OpenRAN como <<https://wiki.o-ran-sc.org/display/SIM/Simulated+datasets>>, onde os formatos dos arquivos são customizados.

Observando-se a literatura em geral, não apenas a relacionada a OpenRAN, dentre os muitos formatos de arquivo, há alguns que almejam descrever formas de onda para processamento de sinais. Um formato que vem sendo promovido pela comunidade do GNURadio é o SigMF (vide <<https://sigmf.org/>> e <<https://pubs.gnuradio.org/index.php/grcon/article/view/52>>). O SigMF é adotado neste projeto para armazenar formas de onda de sinais de rádio, para posterior uso de técnicas de AI/ML.

A proposta do repositório é organizar os dados em duas categorias: formato AI/ML-livre e formato AI/ML-O-RAN. Os *datasets* da primeira categoria podem ser usados com arcabouços genéricos de AI/ML, como *scikit-learn*, *Tensorflow* e *Pytorch*. Os da segunda categoria serão disponibilizados em formatos adequados para uso conjunto com ferramentas da comunidade Open RAN.

3 Soluções baseadas em AI/ML

3.1 Organização

Algumas das aplicações de AI/ML para otimização de Open RAN indicadas na PU são: posicionamento de *virtual network functions* (VNFs), auto-organização (SON), *handover* inteligente, políticas de gerenciamento e gerenciamento de recursos de rádio (RRM). Apesar de ainda não terem sido concluídos a rede O-RAN da Fase 1 e os simuladores genéricos, este relatório descreve resultados obtidos com dados oriundos de simuladores *Asim*, customizados para as aplicações de AI/ML. Essa abordagem permite o desenvolvimento em paralelo e acelera a obtenção dos entregáveis do projeto.

3.2 xApp e rApp Desenvolvidas

Os padrões da O-RAN sugerem que haja uma ampla adoção de AI/ML no desenvolvimento das aplicações, tanto xApps quanto rApps. De fato, conforme será descrito a seguir, a própria orquestração dos componentes dos controladores inteligentes pode se beneficiar do uso de estratégias baseadas em AI/ML. Como soluções que funcionam como provas de conceito da adoção de AI/ML em Open RAN, foram desenvolvidas as seguintes aplicações: uma xApp de *handover* que distribui de maneira eficiente os usuários entre as estações bases e que é auxiliada por outra xApp de monitoramento de uso dos recursos; uma rApp de otimização da eficiência energética que interage com as duas xApps e minimiza a quantidade equipamentos ativos enquanto garante os requisitos dos usuários. Essas aplicações estão descritas em detalhe no relatório da Atividade 3.3.

Além das aplicações apresentadas, outras estão em investigação e desenvolvimento, por exemplo, xApp para redes auto-organizáveis e xApp para fatiamento de rede. Adicionalmente, está sendo investigada uma solução baseada em AI/ML para orquestração dos componentes de um *Near-RT* RIC desagregado e distribuído. Uma breve descrição do trabalho em andamento é apresentado a seguir.

Considerando que um *Near-RT* RIC pode ser desagregado e distribuído em um certo número de grupos de componentes, é importante identificar o número específico que depende de aspectos de implementação, tais como a demanda de recursos de cada componente. Por exemplo, neste projeto, consideramos a implementação do *Near-RT* RIC da OSC e optamos por desagregá-la em cinco grupos principais de componentes: Gerenciamento RIC Near-RT (RIC_Man), E2T, SDL/STSL, NIBs, e xApps, conforme ilustrado na Figura 10. Cada nó E2 está conectado a um componente E2T, o qual serve um conjunto de xApps. Em uma RAN, é possível ter centenas e milhares de nós E2, logo, é comum haver necessidade de criar várias instâncias de componentes como E2T e xApps para atender grupos de nós E2, garantindo as latências exigidas pelos laços de controle usados pelas aplicações que executam sobre o *Near-RT* RIC. A quantidade de instâncias a serem criadas e quais nós E2 devem ser agrupados e associados a cada conjunto de componentes replicados pode mudar dinamicamente, devido a diferentes fatores como flutuação da carga na rede e falhas na infraestrutura, por exemplo.

Em Almeida et al. [7], formalizamos esse problema e o resolvemos de maneira exata e também utilizando uma heurística. Atualmente, estamos investigando o desenvolvimento de uma solução baseada em Aprendizado por Reforço Profundo (do inglês, *Deep Reinforcement Learning* – DRL). Para isso, houve uma reformulação do problema usando uma modelagem baseada em Processo de Decisão de Markov (do inglês, *Markov Decision Process* – MDP), ou seja, a definição de estados, ações e recompensas que servem como base para iniciar a construção de um agente inteligente. Estão sendo avaliadas três abordagens para implementação do agente inteligente: baseada em valor (do inglês, *value-based* usando o algoritmo DQN (*Deep Q-Network*), baseada em política (do inglês,

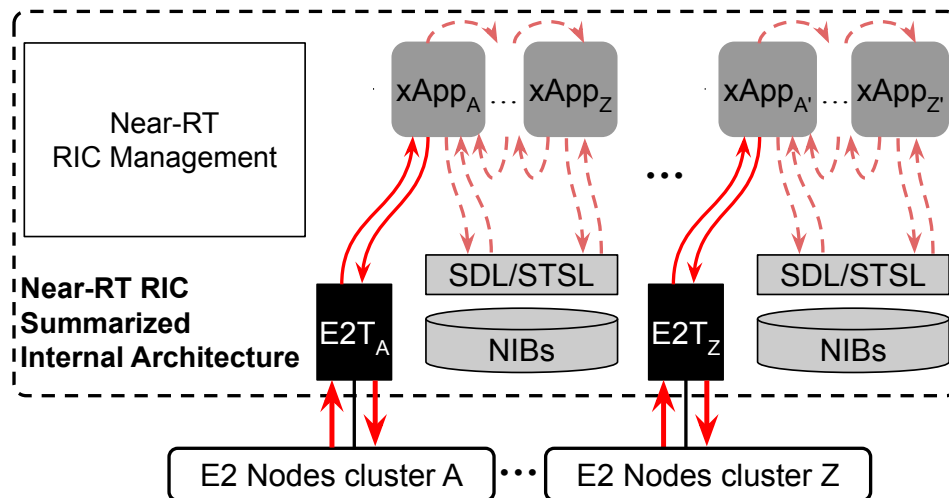


Figura 10 – Versão resumida da arquitetura de componentes do *Near-RT RIC*.

policy-based usando o algoritmo PPO (*Proximal Policy Optimization*) e baseada em Ator-Crítico (do inglês, *actor-critic*) usando o algoritmo A2C (*Advantage Actor Critic*).

3.3 Gerenciamento de recursos de rádio (RRM) para fatiamento da rede

O conceito de fatiamento de rede (NS, do inglês *network slicing*) tem recebido especial interesse ao permitir a criação de múltiplas redes virtuais no topo de uma mesma estrutura de rede física compartilhada [8]. Esse conceito surgiu do recente avanço nas tecnologias de computação e virtualização de funções e possibilita o fornecimento de serviços personalizados para cada cenário de aplicação distinto [9]. O fatiamento da rede de acesso é comumente utilizado para alocação de recursos a fim de garantir o bom desempenho de serviços heterogêneos em 5G. Esses serviços incluem eMBB, mMTC e URLLC [10].

A orquestração do escalonador de recursos de rádio *inter-slice* e *intra-slice* na rede de acesso é essencial para cumprir os requisitos das fatias de rede para os diferentes serviços de aplicações eMBB, mMTC e URLLC. Desta forma, técnicas de aprendizado de máquina são empregadas no controle dos alocadores de recursos de rádio *inter-slice*

e *intra-slice* para atender aos requisitos de qualidade de serviço (QoS, do inglês *quality-of-service*) de cada fatia de rede, com o intuito de evitar violações ao contrato de serviço (SLA, do inglês *service-level agreement*).

Neste trabalho, é proposto a utilização de agentes de aprendizado por reforço para executar as funções de escalonador *inter-slice* e *intra-slice* em uma rede com fatias de rede para aplicações eMBB, mMTC e URLLC. A fatia de rede URLLC é considerada prioritária por normalmente estar associada à aplicações críticas, e por isso possui prioridade em relação a fatias de rede eMBB e mMTC. Logo, caso a quantidade de recursos de rádio disponíveis na rede não sejam suficientes para atender aos requisitos de todas as fatias de rede, o escalonador deve primeiramente garantir os recursos necessários para a fatia de URLLC, e apenas posteriormente tentar garantir os recursos necessários para as fatias remanescentes.

O agente de *inter-slice* é desenvolvido como um rApp e o de *intra-slice* como um xApp, de forma que o rApp é responsável por definir a quantidade de blocos de recurso de rádio que serão alocados para cada fatia de rede, enquanto o agente de *intra-slice* é responsável por escolher um algoritmo para o escalonador dentre as opções *round-robin*, *proportional fair* e *maximum throughput*. Os agentes calculam a sua recompensa utilizando a distância para cumprir os requisitos de cada fatia de rede, de forma que quanto mais distante de cumprir os requisitos, menor é o valor da recompensa. As recompensas das fatias de rede mMTC e eMBB são computadas somente quando os requisitos da fatia de URLLC são atendidos, de forma que os agentes aprendem primeiramente a cumprir os requisitos da fatia URLLC.

A Figura 11 mostra a taxa de transmissão média alcançada por cada uma das fatias da rede. O requisito definido para a taxa de transmissão para a fatia de URLLC e eMBB são 5 Mbps e 20 Mbps. A fatia de mMTC não possui requisito de taxa de transmissão. A figura mostra que a quantidade de recursos disponíveis na rede não são suficientes dado que nenhum dos métodos consegue fornecer os 20 Mbps requisitados pela fatia de eMBB. Os requisitos de URLLC foram fornecidos por ambos os métodos.

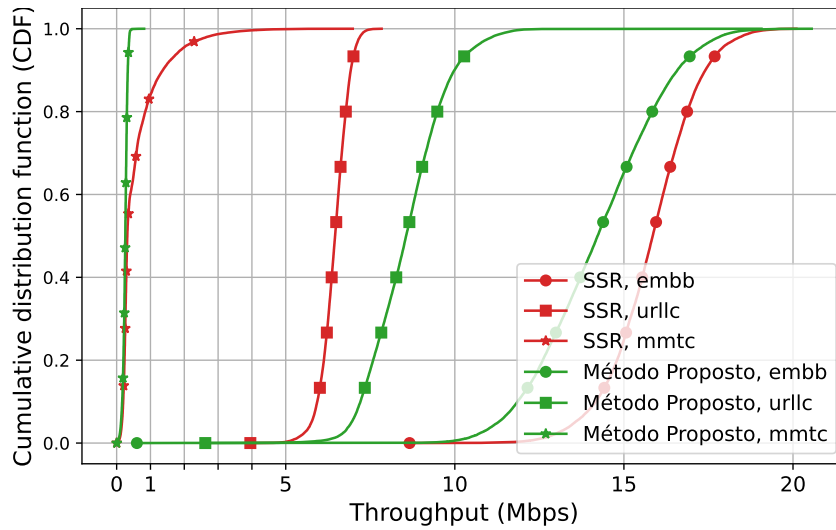


Figura 11 – Taxa de transferência obtida por cada fatia de rede pelo método proposto e o *baseline* SSR. A fatia de URLLC possui requisito de 5 Mbps e a fatia de eMBB 20 Mbps. A fatia de mMTC não possui requisito definido para taxa de transferência.

A Figura 12 mostra a taxa de atraso médio no *buffer* em cada uma das fatias da rede. O requisito definido para o atraso médio no *buffer* para a fatia de URLLC, eMBB e mMTC são 1 ms, 30 ms e 50 ms. Os resultados mostram que o método proposto cumpriu os requisitos de latência para a fatia de URLLC (crítica) pela maior parte do tempo em comparação ao *baseline*. Ambos os métodos conseguem cumprir os requisitos para a fatia de eMBB, enquanto que os requisitos para o mMTC são cumpridos apenas pelo método proposto.

A Figura 13 mostra a comparação da quantidade de violações aos requisitos definidos para todas as fatias de rede e a fatia prioritária de URLLC utilizando o método de *baseline SLA satisfaction rate* (SSR) [11] e o método proposto. A figura representa o valor médio durante 30 episódios com 1000 passos de simulação (TTI), onde os escalonadores tomam uma decisão por TTI. Os resultados demonstram o melhor desempenho do método proposto na proteção a fatia prioritária de URLLC, além do melhor desempenho considerando o número total de violações a todas as fatias de rede.

Os resultados obtidos acima utilizam simulações implementadas em Python

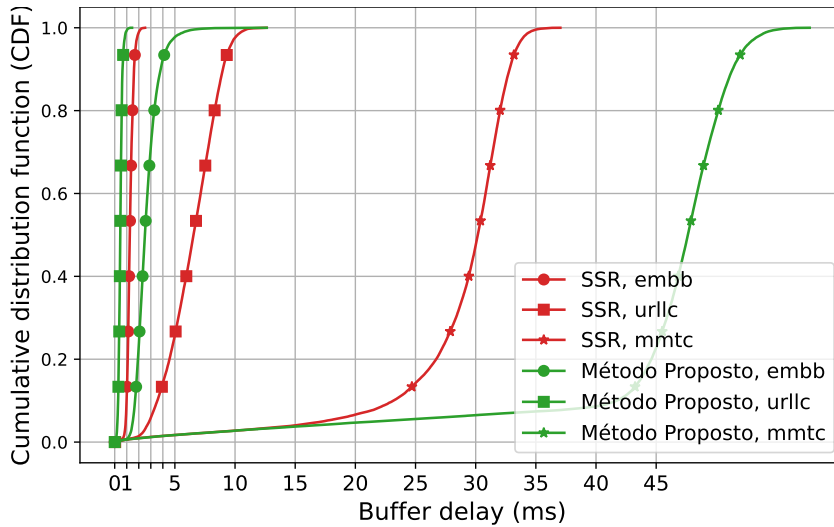


Figura 12 – Latência no buffer média para cada fatia de rede. A fatia de URLLC, eMBB e mMTC possuem requisitos de 1 ms, 30 ms e 50 ms.

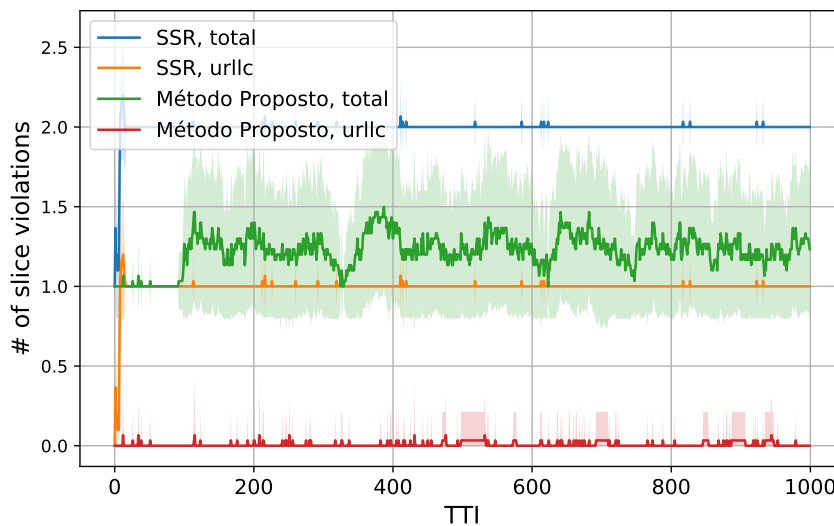


Figura 13 – Número de violações totais e nas fatias de URLLC ao utilizar os métodos de SSR e o método proposto.

em conjunto com o gerador de canais QuaDRiGa [12]. A implementação do método proposto utilizando xApps e rApp na arquitetura Open RAN será realizada no *testbed* proposto pela fase 1 assim que o mesmo for disponibilizado. Nesse momento, está em progresso a implementação do cenário no simulador ns-3 com integração ao controlador *Near-RT RIC*.

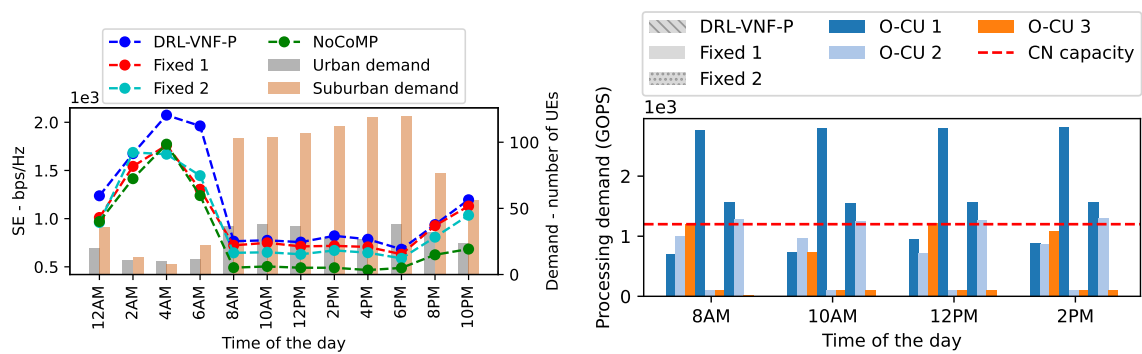
3.4 Aprendizado por Reforço Aplicado a Posicionamento de VNFs

De forma similar à aplicação discutida na seção anterior deste relatório, os próximos parágrafos descrevem uma aplicação de IA/ML baseada em aprendizado por reforço. Neste caso, apresenta-se o uso de *deep reinforcement learning* aplicado ao posicionamento de VNFs.

A mitigação de interferências é um benefício comum reivindicado por redes Open RAN desagregadas e virtualizadas (vRAN abertas). No entanto, esse benefício depende do posicionamento adequado das Funções de Rede Virtuais (VNFs) da pilha de protocolos das unidades de rádio (RUs) vizinhas. Adicionalmente, os recursos computacionais disponíveis e a demanda dinâmica da rede devem ser levados em consideração para a obtenção de resultados eficientes. A Figura 14 apresenta uma visão dos nós e da rede de transporte, além de uma visão da área de cobertura das estações bases (ou RUs). Além disso, são ilustrados nessa figura os *clusters* que criados para aplicar uma técnica de mitigação de interferência chamada CoMP (*Coordinated Multi-Point*).

Em Lopes et al. [13], formulamos o problema descrito como um Processo de Decisão de Markov (MDP) e o resolvemos empregando um agente de Aprendizado por Reforço Profundo (DRL). Além disso, descrevemos como nossa proposta pode ser implementada dentro da arquitetura O-RAN, conforme ilustrado na Figura 15. Uma solução de posicionamento de VNFs estabelece: 1) para cada O-RU (O-RAN RU) qual é a divisão funcional da pilha selecionada, 2) em qual nó cada parte da divisão funcional será executada e quais enlaces serão usados. Essa solução só pode ser estabelecida se

flutuações de carga na rede ao longo de um dia, usando dados de uma infraestrutura real. A solução baseada em AI/ML foi comparada com duas abordagens tradicionais que chamamos de *Fixed 1* (*cluster* único) e *Fixed 2* (dois *clusters*). Conforme ilustrado nos gráficos da Figura 16, a estratégia baseada em AI/ML apresenta maior eficiência espectral na maior parte do tempo (Figura 16a), enquanto garante que os recursos computacionais não são sobrecarregados (Figura 16b). As consequências desse tipo de sobrecargas seriam severas, sobretudo para aplicações mais sensíveis a perdas e atrasos.



(a) Eficiência espectral obtida pro cada estratégia ao longo do dia. (b) Demanda por recursos computacionais para processamento de sinais.

Figura 16 – Eficiência espectral e demanda por recursos computacionais sobre diferentes condições de carga na rede.

4 Conclusão

Este relatório apresentou como a adoção de AI/ML é relevante tanto no contexto de aplicações O-RAN, i.e., xApps e rApps, quanto de toda a arquitetura O-RAN. Foram mostrados benefícios em termos de desempenho e uso de recursos que justificam os esforços que vem sendo realizados para uma utilização efetiva de AI/ML em Open RAN. O relatório apresentou diversos exemplos do uso de AI/ML em Open RAN desenvolvidos até o momento no projeto, sendo alguns concluídos e outros ainda em evolução. Acredita-se que a continuidade dessa evolução é uma contribuição relevante para o projeto, uma vez que documentação e código estão sendo gerados e disponibilizados para a comunidade brasileira de telecomunicações, contribuindo com a formação de profissionais qualificados e atualizados com as tecnologias de redes Pós-5G, como Open RAN.

5 Referências

- 1 LACAVA, A.; BORDIN, M.; POLESE, M.; SIVARAJ, R.; ZUGNO, T.; CUOMO, F.; MELODIA, T. ns-O-RAN: Simulating O-RAN 5G Systems in ns-3. In: *Proceedings of the 2023 Workshop on ns-3*. ACM, 2023. (WNS3 2023). Disponível em: <<http://dx.doi.org/10.1145/3592149.3592161>>. Citado 7 vezes nas páginas 3, 11, 12, 13, 14, 15 e 16.
- 2 FERREIRA, G.; BARRETO, P.; ALCHIERI, E.; CARVALHO, P. ns3-ORAN: Uma Implementação do Open-RAN para o Simulador ns-3. In: *Anais Estendidos do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2023. p. 24–31. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/24629>. Citado 4 vezes nas páginas 3, 11, 12 e 13.
- 3 HENDERSON, T. R. Network simulations with the ns-3 simulator. In: . [s.n.], 2008. Disponível em: <<https://api.semanticscholar.org/CorpusID:18689058>>. Citado na página 11.
- 4 ZUGNO, T.; POLESE, M.; PATRICIELLO, N.; BOJOVIĆ, B.; LAGEN, S.; ZORZI, M. Implementation of a spatial channel model for ns-3. In: *Proceedings of the 2020 Workshop on ns-3*. ACM, 2020. (WNS3 2020). Disponível em: <<http://dx.doi.org/10.1145/3389400.3389401>>. Citado na página 18.
- 5 PAGIN, M.; LAGÉN, S.; BOJOVIC, B.; POLESE, M.; ZORZI, M. Improving the efficiency of MIMO simulations in ns-3. In: *Proceedings of the 2023 Workshop on ns-3*. [S.l.: s.n.], 2023. Citado na página 18.
- 6 LECCI, M.; TESTOLINA, P.; POLESE, M.; GIORDANI, M.; ZORZI, M. *Accuracy vs. Complexity for mmWave Ray-Tracing: A Full Stack Perspective*. 2020. Disponível em: <<https://arxiv.org/abs/2007.07125>>. Citado na página 19.
- 7 ALMEIDA, G. M.; BRUNO, G. Z.; HUFF, A.; HILTUNEN, M.; DUARTE, E. P.; BOTH, C. B.; CARDOSO, K. V. RIC-O: Efficient Placement of a Disaggregated and Distributed RAN Intelligent Controller With Dynamic Clustering of Radio Nodes. *IEEE Journal on Selected Areas in Communications*, v. 42, n. 2, p. 446–459, 2024. Citado na página 25.
- 8 ALWIS, C. D. et al. A survey on network slicing security: Attacks, challenges, solutions and research directions. *IEEE Communications Surveys & Tutorials*, v. 26, n. 1, p. 534–570, Firstquarter 2024. Citado na página 26.

- 9 SALAHDINE, F.; LIU, Q.; HAN, T. Towards secure and intelligent network slicing for 5G networks. *IEEE Open Journal of the Computer Society*, v. 3, p. 23–38, Mar. 2022. Citado na página 26.
- 10 LIU, Y. et al. Network slicing for eMBB, URLLC, and mMTC: An uplink rate-splitting multiple access approach. *IEEE Transactions on Wireless Communications*, v. 23, n. 3, p. 2140–2152, Mar. 2024. Citado na página 26.
- 11 LI, R. et al. The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility. *IEEE Communications Letters*, IEEE, v. 24, n. 9, p. 2005–2009, 2020. Citado na página 28.
- 12 JAECKEL, S. et al. Industrial Indoor Measurements from 2-6 GHz for the 3GPP-NR and QuaDRiGa Channel Model. In: *IEEE 90th Vehicular Technology Conference*. [S.l.: s.n.], 2019. p. 1–7. Citado na página 30.
- 13 LOPES, V. H. L.; ALMEIDA, G. M.; KLAUTAU, A.; CARDOSO, K. O-RAN-oriented approach for dynamic VNF placement focused on interference mitigation. In: IEEE. *2024 IEEE International Conference on Communications (ICC)*. [S.l.], 2024. p. 1–6. Citado na página 30.

6 Histórico de versões deste documento

<u>Data de Emissão</u>	<u>Versão</u>	<u>Descrição das Alterações Realizadas</u>
10/07/2024	1.0	Construção inicial do relatório
DD/MM/2024	1.?	Revisão de todo documento

7 Execução e aprovação

Executado por:

Luan Rios

Revisado por:

Fernando Farias

Aprovado por:

Fernando Farias

Data da emissão: DD/MM/2024